EXHIBIT 7

International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.265

(08/2021)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving video

---

High efficiency video coding

Recommendation ITU-T H.265

ITU-T

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	**H.260–H.279**
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Telepresence, immersive environments, virtual and extended reality	H.420–H.439
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
VEHICULAR GATEWAYS AND INTELLIGENT TRANSPORTATION SYSTEMS (ITS)	
Architecture for vehicular gateways	H.550–H.559
Vehicular gateway interfaces	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	H.620–H.629
Content delivery and ubiquitous sensor network applications	H.640–H.649
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	H.760–H.769
IPTV service discovery up to consumption	H.770–H.779
Digital Signage	H.780–H.789
E-HEALTH MULTIMEDIA SYSTEMS, SERVICES AND APPLICATIONS	
Personal health systems	H.810–H.819
Interoperability compliance testing of personal health systems (HRN, PAN, LAN, TAN and WAN)	H.820–H.859
Multimedia e-health data exchange services	H.860–H.869
Safe listening	H.870–H.879

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T H.265

High efficiency video coding

Summary

Recommendation ITU-T H.265 | International Standard ISO/IEC 23008-2 represents an evolution of the existing video coding Recommendations (ITU-T H.261, ITU-T H.262, ITU-T H.263 and ITU-T H.264) and was developed in response to the growing need for higher compression of moving pictures for various applications such as Internet streaming, communication, videoconferencing, digital storage media and television broadcasting. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

This revision adds an additional SEI message for shutter interval information, and also includes corrections to various minor defects in the prior content of the Specification.

This Recommendation | International Standard was developed jointly with ISO/IEC JTC 1/SC 29 and corresponds in a technically aligned manner to ISO/IEC 23008-2.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T H.265	2013-04-13	16	11.1002/1000/11885
2.0	ITU-T H.265 (V2)	2014-10-29	16	11.1002/1000/12296
3.0	ITU-T H.265 (V3)	2015-04-29	16	11.1002/1000/12455
4.0	ITU-T H.265 (V4)	2016-12-22	16	11.1002/1000/12905
5.0	ITU-T H.265 (V5)	2018-02-13	16	11.1002/1000/13433
6.0	ITU-T H.265 (V6)	2019-06-29	16	11.1002/1000/13904
7.0	ITU-T H.265 (V7)	2019-11-29	16	11.1002/1000/14107
8.0	ITU-T H.265 (V8)	2021-08-22	16	11.1002/1000/14660

Keywords

Compression coding, digital video, image coding, supplemental enhancement information, video coding, video compression, visual coding.

* To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Foreword

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a world-wide basis. The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups that, in turn, produce Recommendations on these topics. The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1. In some areas of information technology that fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for world-wide standardization. National Bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

This Recommendation | International Standard was prepared jointly by ITU-T SG 16 Question 6/16, also known as VCEG (Video Coding Experts Group), and by the working groups of ISO/IEC JTC 1/SC 29 known as MPEG (Moving Picture Experts Group). VCEG was formed in 1997 to maintain prior ITU-T video coding standards and develop new video coding standard(s) appropriate for a wide range of conversational and non-conversational services. MPEG was formed in 1988 to establish standards for coding of moving pictures and associated audio for various applications such as digital storage media, distribution, and communication.

In this Recommendation | International Standard Annexes A through I contain normative requirements and are an integral part of this Recommendation | International Standard.

Recommendation ITU-T H.265

High efficiency video coding

0 Introduction

0.1 General

This clause and its subclauses do not form an integral part of this Recommendation | International Standard.

0.2 Prologue

As the costs for both processing power and memory have reduced, network support for coded video data has diversified, and advances in video coding technology have progressed, the need has arisen for an industry standard for compressed video representation with substantially increased coding efficiency and enhanced robustness to network environments. Toward these ends the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) formed a Joint Collaborative Team on Video Coding (JCT-VC) in 2010 and a Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) in 2012 for development of a new Recommendation | International Standard. This Recommendation | International Standard was developed in the JCT-VC and the JCT-3V until 2020, when the responsibility for further maintenance and enhancement of the standard was transferred to another joint collaborative team of the same organizations called the Joint Video Experts Team (JVET).

0.3 Purpose

This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming, and communications. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments as well as to enable the use of multi-core parallel encoding and decoding devices. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels. Supports for higher bit depths and enhanced chroma formats, including the use of full-resolution chroma are provided. Support for scalability enables video transmission on networks with varying transmission conditions and other scenarios involving multiple bit rate services. Support for multiview enables representation of video content with multiple camera views and optional auxiliary information. Support for 3D enables joint representation of video content and depth information with multiple camera views.

0.4 Applications

This Recommendation | International Standard is designed to cover a broad range of applications for video content including but not limited to the following:

- Broadcast (cable TV on optical networks / copper, satellite, terrestrial, etc.)
- Camcorders
- Content production and distribution
- Digital cinema
- Home cinema
- Internet streaming, download and play
- Medical imaging
- Mobile streaming, broadcast and communications
- Real-time conversational services (videoconferencing, videophone, telepresence, etc.)
- Remote video surveillance
- Storage media (optical disks, digital video tape recorder, etc.)
- Wireless display

0.5 Publication and versions of this Specification

This Specification has been jointly developed by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). It is published as technically-aligned twin text in both ITU-T and ISO/IEC. As the basis text has been drafted to become both an ITU-T Recommendation and an ISO/IEC International Standard, the term "Specification" (with capitalization to indicate that it refers to the whole of the text) is used herein when the text refers to itself.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 1 refers to the first approved version of this Recommendation | International Standard. The first edition published by ITU-T as Rec. ITU-T H.265 (04/2013) and by ISO/IEC as ISO/IEC 23008-2:2013 corresponded to the first version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 2 refers to the integrated text additionally containing format range extensions, scalability extensions, multiview extensions, additional supplement enhancement information, and corrections to various minor defects in the prior content of the Specification. The second edition published by ITU-T as Rec. H.265 (10/2014) and by ISO/IEC as ISO/IEC 23008-2:2015 corresponded to the second version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 3 refers to the integrated text additionally containing 3D extensions, additional supplement enhancement information, and corrections to various minor defects in the prior content of the Specification. The third edition published by ITU-T as Rec. H.265 (04/2015) corresponded to the third version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 4 refers to the integrated text additionally containing screen content coding extensions profiles, scalable range extensions profiles, additional high throughput profiles, additional supplement enhancement information, additional colour representation identifiers, and corrections to various minor defects in the prior content of the Specification. The fourth edition published by ITU-T as Rec. H.265 (12/2016) and by ISO/IEC as ISO/IEC 23008-2:2017 corresponded to the fourth version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 5 refers to the integrated text additionally containing additional SEI messages that include omnidirectional video SEI messages, a Monochrome 10 profile, a Main 10 Still Picture profile, and corrections to various minor defects in the prior content of the Specification. The fifth edition published by ITU-T as Rec. H.265 (02/2018) corresponded to the fifth version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 6 refers to the integrated text additionally containing additional SEI messages for SEI manifest and SEI prefix, along with some corrections to the existing specification text. The sixth edition published by ITU-T as Rec. H.265 (06/2019) corresponded to the sixth version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 7 refers to the integrated text additionally containing the fisheye video information SEI message and the annotated regions SEI message, along with some corrections to the existing specification text. The seventh edition published by ITU-T as Rec. H.265 (11/2019) corresponded to the seventh version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 8 (the current version) refers to the integrated text additionally containing the shutter interval information SEI message, along with some corrections to the existing specification text.

0.6 Profiles, tiers and levels

This Recommendation | International Standard is designed to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities and services. Applications should cover, among other things, digital storage media, television broadcasting and real-time communications. In the course of creating this Specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and these have been integrated into a single syntax. Hence, this Specification will facilitate video data interchange among different applications.

Considering the practicality of implementing the full syntax of this Specification, however, a limited number of subsets of the syntax are also stipulated by means of "profiles", "tiers" and "levels". These and other related terms are formally defined in clause 3.

A "profile" is a subset of the entire bitstream syntax that is specified in this Recommendation | International Standard. Within the bounds imposed by the syntax of a given profile, it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by syntax elements in the bitstream such as the specified size of the decoded pictures. In many applications, it is currently neither practical nor economical to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile.

In order to deal with this problem, "tiers" and "levels" are specified within each profile. A level of a tier is a specified set of constraints imposed on values of the syntax elements in the bitstream. These constraints may be simple limits on values. Alternatively they may take the form of constraints on arithmetic combinations of values (e.g., picture width multiplied by picture height multiplied by number of pictures decoded per second). A level specified for a lower tier is more constrained than a level specified for a higher tier.

Coded video content conforming to this Recommendation | International Standard uses a common syntax. In order to achieve a subset of the complete syntax, flags, parameters and other syntax elements are included in the bitstream that signal the presence or absence of syntactic elements that occur later in the bitstream.

0.7 Overview of the design characteristics

The coded representation specified in the syntax is designed to enable a high compression capability for a desired image or video quality. The algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes. A number of techniques may be used to achieve highly efficient compression. Encoding algorithms (not specified in this Recommendation | International Standard) may select between inter and intra coding for block-shaped regions of each picture. Inter coding uses motion vectors for block-based inter prediction to exploit temporal statistical dependencies between different pictures. Intra coding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. Motion vectors and intra prediction modes may be specified for a variety of block sizes in the picture. The prediction residual may then be further compressed using a transform to remove spatial correlation inside the transform block before it is quantized, producing a possibly irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes may also be further compressed using a variety of prediction mechanisms, and, after prediction, are combined with the quantized transform coefficient information and encoded using arithmetic coding.

0.8 How to read this Specification

It is suggested that the reader starts with clause 1 (Scope) and moves on to clause 3 (Definitions). Clause 6 should be read for the geometrical relationship of the source, input and output of the decoder. Clause 7 (Syntax and semantics) specifies the order to parse syntax elements from the bitstream. See clauses 7.1–7.3 for syntactical order and see clause 7.4 for semantics; e.g., the scope, restrictions and conditions that are imposed on the syntax elements. The actual parsing for most syntax elements is specified in clause 9 (Parsing process). Clause 10 (Sub-bitstream extraction process) specifies the sub-bitstream extraction process. Finally, clause 8 (Decoding process) specifies how the syntax elements are mapped into decoded samples. Throughout reading this Specification, the reader should refer to clauses 2 (Normative references), 4 (Abbreviations), and 5 (Conventions) as needed. Annexes A through I also form an integral part of this Recommendation | International Standard.

Annex A specifies profiles each being tailored to certain application domains, and defines the so-called tiers and levels of the profiles. Annex B specifies syntax and semantics of a byte stream format for delivery of coded video as an ordered stream of bytes. Annex C specifies the hypothetical reference decoder, bitstream conformance, decoder conformance and the use of the hypothetical reference decoder to check bitstream and decoder conformance. Annex D specifies syntax and semantics for supplemental enhancement information message payloads. Annex E specifies syntax and semantics of the video usability information parameters of the sequence parameter set. Annex F specifies general multi-layer support for bitstreams and decoders. Annex G contains support for multiview coding. Annex H contains support for scalability. Annex I contains support for 3D coding.

Throughout this Specification, statements appearing with the preamble "NOTE –" are informative and are not an integral part of this Recommendation | International Standard.

1 Scope

This Recommendation | International Standard specifies high efficiency video coding.

2 Normative references

2.1 General

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.2 Identical Recommendations | International Standards

– None

2.3 Paired Recommendations | International Standards equivalent in technical content

– None

2.4 Additional references

- Recommendation ITU-T T.35 (in force), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*.
- ISO/IEC 10646: in force, *Information technology – Universal Coded Character Set (UCS)*.
- ISO/IEC 11578: in force, *Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)*.
- ISO 11664-1: in force, *Colorimetry – Part 1: CIE standard colorimetric observers*.
- ISO 12232: in force, *Photography – Digital still cameras – Determination of exposure index, ISO speed ratings, standard output sensitivity, and recommended exposure index*.
- IETF RFC 1321 (in force), *The MD5 Message-Digest Algorithm*.
- IETF RFC 5646 (in force), *Tags for Identifying Languages.*
- ISO/IEC 23001-11 (in force), *Information Technology – MPEG Systems technologies – Part 11: Energy-efficient media consumption (green metadata).*

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply:

3.1 **access unit**: A set of *NAL units* that are associated with each other according to a specified classification rule, are consecutive in *decoding order,* and contain exactly one *coded picture* with nuh_layer_id equal to 0.

NOTE 1 – In addition to containing the video coding layer (VCL) NAL units of the coded picture with nuh_layer_id equal to 0, an access unit may also contain non-VCL NAL units. The decoding of an access unit with the decoding process specified in clause 8 always results in a decoded picture with nuh_layer_id equal to 0.

NOTE 2 – An access unit is defined differently in Annex F and does not need to contain a coded picture with nuh_layer_id equal to 0.

3.2 **AC transform coefficient**: Any *transform coefficient* for which the *frequency index* in at least one of the two dimensions is non-zero.

3.3 **associated IRAP picture**: The previous *IRAP picture* in *decoding order* (when present).

3.4 **associated non-VCL NAL unit**: A *non-VCL NAL unit* (when present) for a *VCL NAL unit* where the *VCL NAL unit* is the *associated VCL NAL unit* of the *non-VCL NAL unit*.

3.5 **associated VCL NAL unit**: The preceding *VCL NAL unit* in *decoding order* for a *non-VCL NAL unit* with nal_unit_type equal to EOS_NUT, EOB_NUT, FD_NUT or SUFFIX_SEI_NUT, or in the ranges of RSV_NVCL45..RSV_NVCL47 or UNSPEC56..UNSPEC63; or otherwise the next *VCL NAL unit* in *decoding order*.

3.6 **azimuth circle**: A circle on a sphere connecting all points with the same azimuth value.

NOTE – An azimuth circle is always a *great circle* like a longitude line on the earth.

3.7 **base layer**: A *layer* in which all *NAL units* have nuh_layer_id equal to 0.

3.8 **bin**: One bit of a *bin string*.

3.9 **binarization**: A set of *bin strings* for all possible values of a *syntax element*.

3.10 **binarization process**: A unique mapping process of all possible values of a *syntax element* onto a set of *bin strings*.

3.11 **bin string**: An intermediate binary representation of values of *syntax elements* from the *binarization* of the *syntax element*.

3.12 **bi-predictive (B) slice**: A *slice* that is decoded using *intra prediction* or using *inter prediction* with at most two *motion vectors* and *reference indices* to *predict* the sample values of each *block*.

3.13 **bitstream**: A sequence of bits, in the form of a *NAL unit stream* or a *byte stream*, that forms the representation of *coded pictures* and associated data forming one or more coded video sequences *(CVSs)*.

3.14 **block**: An MxN (M-column by N-row) array of samples, or an MxN array of *transform coefficients*.

3.15 **broken link**: A location in a *bitstream* at which it is indicated that some subsequent *pictures* in *decoding order* may contain serious visual artefacts due to unspecified operations performed in the generation of the *bitstream*.

3.16 **broken link access (BLA) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is a *BLA picture*.

3.17 **broken link access (BLA) picture**: An *IRAP picture* for which each *VCL NAL unit* has nal_unit_type equal to BLA_W_LP, BLA_W_RADL, or BLA_N_LP.

NOTE – A BLA picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each BLA picture begins a new CVS, and has the same effect on the decoding process as an instantaneous decoding refresh (IDR) picture. However, a BLA picture contains syntax elements that specify a non-empty RPS. When a BLA picture for which each VCL NAL unit has nal_unit_type equal to BLA_W_LP, it may have associated random access skipped leading (RASL) pictures, which are not output by the decoder and may not be decodable, as they may contain references to pictures that are not present in the bitstream. When a BLA picture for which each VCL NAL unit has nal_unit_type equal to BLA_W_LP, it may also have associated RADL pictures, which are specified to be decoded. When a BLA picture for which each VCL NAL unit has nal_unit_type equal to BLA_W_RADL, it does not have associated RASL pictures but may have associated random access decodable leading (RADL) pictures. When a BLA picture for which each VCL NAL unit has nal_unit_type equal to BLA_N_LP, it does not have any associated leading pictures.

3.18 **buffering period**: The set of *access units* starting with an *access unit* that contains a buffering period suplemental enhancement information (SEI) message and containing all subsequent *access units* in *decoding order* up to but not including the next *access unit* (when present) that contains a buffering period SEI message.

3.19 **byte**: A sequence of 8 bits, within which, when written or read as a sequence of bit values, the left-most and right-most bits represent the most and least significant bits, respectively.

3.20 **byte-aligned**: A position in a *bitstream* is byte-aligned when the position is an integer multiple of 8 bits from the position of the first bit in the *bitstream*, and a bit or *byte* or *syntax element* is said to be byte-aligned when the position at which it appears in a *bitstream* is byte-aligned.

3.21 **byte stream**: An encapsulation of a *NAL unit stream* containing *start code prefixes* and *NAL units* as specified in Annex B.

3.22 **can**: A term used to refer to behaviour that is allowed, but not necessarily required.

3.23 **chroma**: An adjective, represented by the symbols Cb and Cr, specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours.

NOTE – The term chroma is used rather than the term chrominance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term chrominance.

3.24 **clean random access (CRA) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is a *CRA picture*.

3.25 **clean random access (CRA) picture**: An *IRAP picture* for which each *VCL NAL unit* has nal_unit_type equal to CRA_NUT.

NOTE – A CRA picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. A CRA picture may have associated RADL or RASL pictures. As with a BLA picture, a CRA picture may contain syntax elements that specify a non-empty RPS. When a CRA picture has NoRaslOutputFlag equal to 1, the associated RASL pictures are not output by the decoder, because they may not be decodable, as they may contain references to pictures that are not present in the bitstream.

3.26 **coded picture**: A *coded representation* of a *picture* containing all *CTUs* of the *picture*.

3.27 **coded picture buffer (CPB)**: A first-in first-out buffer containing *decoding units* in *decoding order* specified in the *hypothetical reference decoder* in Annex C.

3.28 **coded representation**: A data element as represented in its coded form.

3.29 **coded slice segment NAL unit**: A *NAL unit* that has nal_unit_type in the range of TRAIL_N to RASL_R, inclusive, or in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, which indicates that the *NAL unit* contains a coded *slice segment*.

3.30 **coded layer-wise video sequence (CLVS)**: A sequence of *pictures* and the *associated non-VCL NAL units* of the *base layer* of a *coded video sequence* (*CVS*).

3.31 **coded video sequence (CVS)**: A sequence of *access units* that consists, in *decoding order*, of an *IRAP access unit* with NoRaslOutputFlag equal to 1, followed by zero or more *access units* that are not *IRAP access units* with NoRaslOutputFlag equal to 1, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *IRAP access unit* with NoRaslOutputFlag equal to 1.

NOTE – An IRAP access unit may be an IDR access unit, a BLA access unit, or a CRA access unit. The value of NoRaslOutputFlag is equal to 1 for each IDR access unit, each BLA access unit, and each CRA access unit that is the first access unit in the bitstream in decoding order, is the first access unit that follows an end of sequence NAL unit in decoding order, or has HandleCraAsBlaFlag equal to 1.

3.32 **coded video sequence group (CVSG)**: One or more consecutive *CVSs* in *decoding order* that collectively consist of an *IRAP access unit* that activates a video parameter set (VPS) RBSP firstVpsRbsp that was not already active followed by all subsequent *access units*, in *decoding order*, for which firstVpsRbsp is the active VPS raw byte sequence payload (RBSP) up to the end of the *bitstream* or up to but excluding the *access unit* that activates a different VPS RBSP than firstVpsRbsp, whichever is earlier in *decoding order*.

3.33 **coding block**: An NxN *block* of samples for some value of N such that the division of a *CTB* into *coding blocks* is a *partitioning*.

3.34 **coding tree block (CTB)**: An NxN *block* of samples for some value of N such that the division of a *component* into *CTBs* is a *partitioning*.

3.35 **coding tree unit (CTU)**: A *CTB* of *luma* samples, two corresponding *CTBs* of *chroma* samples of a *picture* that has three sample arrays, or a *CTB* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes and *syntax structures* used to code the samples.

3.36 **coding unit**: A *coding block* of *luma* samples, two corresponding *coding blocks* of *chroma* samples of a *picture* that has three sample arrays, or a *coding block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes and *syntax structures* used to code the samples.

3.37 **component**: An array or single sample from one of the three arrays (*luma* and two *chroma*) that compose a *picture* in 4:2:0, 4:2:2, or 4:4:4 colour format or the array or a single sample of the array that compose a *picture* in monochrome format.

3.38 **constituent picture**: A part of a spatially *frame* -packed stereoscopic *picture* that corresponds to one view, or a *picture* itself when *frame* packing is not in use or the temporal interleaving *frame* packing arrangement is in use.

3.39 **context variable**: A variable specified for the *adaptive binary arithmetic decoding process* of a *bin* by an equation containing recently decoded *bins*.

3.40 **cropped decoded picture**: The result of cropping a *decoded picture* based on the conformance cropping window specified in the *sequence parameter set (SPS)* that is referred to by the corresponding *coded picture*.

3.41 **decoded picture**: A *decoded picture* is derived by decoding a *coded picture*.

3.42 **decoded picture buffer (DPB)**: A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C.

3.43 **decoder**: An embodiment of a *decoding process*.

3.44 **decoder under test (DUT)**: A *decoder* that is tested for conformance to this Specification by operating the *hypothetical stream scheduler* to deliver a conforming *bitstream* to the *decoder* and to the *hypothetical reference decoder* and comparing the values and timing or order of the output of the two *decoders*.

3.45 **decoding order**: The order in which *syntax elements* are processed by the *decoding process*.

3.46 **decoding process**: The process specified in this Specification that reads a *bitstream* and derives *decoded pictures* from it.

3.47 **decoding unit**: An *access unit* if SubPicHrdFlag is equal to 0 or a subset of an *access unit* otherwise, consisting of one or more *VCL NAL units* in an *access unit* and the *associated non-VCL NAL units*.

3.48 **dependent slice segment**: A *slice segment* for which the values of some *syntax elements* of the *slice segment header* are inferred from the values for the preceding *independent slice segment* in *decoding order*.

3.49 **display process**: A process not specified in this Specification having, as its input, the *cropped decoded pictures* that are the output of the *decoding process*.

3.50 **elementary stream**: A sequence of one or more *bitstreams*.

NOTE – An elementary stream that consists of two or more bitstreams would typically have been formed by splicing together two or more bitstreams (or parts thereof).

3.51 **elevation circle**: A circle on a sphere connecting all points with the same elevation value.

NOTE – An elevation circle is similar to a lattitude line on the earth. Except when the elevation value is zero, an elevation circle is not a *great circle* like a longitude circle on the earth.

3.52 **emulation prevention byte**: A *byte* equal to 0x03 that is present within a *NAL unit* when the *syntax elements* of the *bitstream* form certain patterns of *byte* values in a manner that ensures that no sequence of consecutive *byte-aligned bytes* in the *NAL unit* can contain a *start code prefix*.

3.53 **encoder**: An embodiment of an *encoding process*.

3.54 **encoding process**: A process not specified in this Specification that produces a *bitstream* conforming to this Specification.

3.55 **field**: An assembly of alternative rows of samples of a *frame*.

3.56 **filler data NAL units**: *NAL units* with nal_unit_type equal to FD_NUT.

3.57 **flag**: A variable or single-bit *syntax element* that can take one of the two possible values: 0 and 1.

3.58 **frame**: The composition of a top *field* and a bottom *field*, where sample rows 0, 2, 4, ... originate from the top *field* and sample rows 1, 3, 5, ... originate from the bottom *field*.

3.59 **frequency index**: A one-dimensional or two-dimensional index associated with a *transform coefficient* prior to the application of a *transform* in the *decoding process*.

3.60 **global coordinate axes**: The coordinate axes associated with *omnidirectional video* that are associated with an externally referenceable position and orientation.

NOTE – The global coordinate axes may correspond to the position and orientation of a device or rig used for omnidirectional audio/video acquisition as well as the position of an observer's head in the three-dimensional space of the *omnidirectional video* rendering environment.

3.61 **great circle**: The intersection of a sphere and a plane that passes through the centre point of the sphere.

NOTE – A great circle is also known as an orthodrome or Riemannian circle.

3.62 **hypothetical reference decoder (HRD)**: A hypothetical *decoder* model that specifies constraints on the variability of conforming *NAL unit streams* or conforming *byte streams* that an encoding process may produce.

3.63 **hypothetical stream scheduler (HSS)**: A hypothetical delivery mechanism used for checking the conformance of a *bitstream* or a *decoder* with regards to the timing and data flow of the input of a *bitstream* into the *hypothetical reference decoder*.

3.64 **independent slice segment**: A *slice segment* for which the values of the *syntax elements* of the *slice segment header* are not inferred from the values for a preceding *slice segment*.

3.65 **informative**: A term used to refer to content provided in this Specification that does not establish any mandatory requirements for conformance to this Specification and thus is not considered an integral part of this Specification.

3.66 **instantaneous decoding refresh (IDR) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is an *IDR picture*.

3.67 **instantaneous decoding refresh (IDR) picture**: An *IRAP picture* for which each *VCL NAL unit* has nal_unit_type equal to IDR_W_RADL or IDR_N_LP.

NOTE – An IDR picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each IDR picture is the first picture of a CVS in decoding order. When an IDR picture for which each VCL NAL unit has nal_unit_type equal to IDR_W_RADL, it may have associated RADL pictures. When an IDR picture for which each VCL NAL unit has nal_unit_type equal to IDR_N_LP, it does not have any associated leading pictures. An IDR picture does not have associated RASL pictures.

3.68 **inter coding**: Coding of a *coding block*, *slice*, or *picture* that uses *inter prediction*.

3.69 **inter prediction**: A *prediction* derived in a manner that is dependent on data elements (e.g., sample values or motion vectors) of one or more *reference pictures*.

NOTE – A prediction from a reference picture that is the current picture itself is also inter prediction.

3.70 **intra coding**: Coding of a *coding block*, *slice*, or *picture* that uses *intra prediction*.

3.71 **intra prediction**: A *prediction* derived from only data elements (e.g., sample values) of the same decoded *slice* without referring to a *reference picture*.

3.72 **intra random access point (IRAP) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is an *IRAP picture*.

3.73 **intra random access point (IRAP) picture**: A coded *picture* for which each *VCL NAL unit* has nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.

NOTE – An IRAP picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be a BLA picture, a CRA picture or an IDR picture. The first picture in the bitstream in decoding order must be an IRAP picture. Provided the necessary parameter sets are available when they need to be activated, the IRAP picture and all subsequent non-RASL pictures in decoding order can be correctly decoded without performing the decoding process of any pictures that precede the IRAP picture in decoding order. There may be pictures in a bitstream that do not refer to any pictures other than itself for inter prediction in its decoding process that are not IRAP pictures.

3.74 **intra (I) slice**: A *slice* that is decoded using *intra prediction* only.

3.75 **layer**: A set of *VCL NAL units* that all have a particular value of nuh_layer_id and the *associated non-VCL NAL units*, or one of a set of syntactical structures having a hierarchical relationship.

NOTE – Depending on the context, either the first layer concept or the second layer concept applies. The first layer concept is also referred to as a scalable layer, wherein a layer may be a spatial scalable layer, a quality scalable layer, a view, etc. A temporal true subset of a scalable layer is not referred to as a layer but referred to as a sub-layer or temporal sub-layer. The second layer concept is also referred to as a coding layer, wherein higher layers contain lower layers, and the coding layers are the CVS, picture, slice, slice segment, and CTU layers.

3.76 **layer identifier list**: A list of nuh_layer_id values that is associated with a *layer set* or an *operation point* and can be used as an input to the *sub-bitstream extraction process*.

3.77 **layer set**: A set of *layers* represented within a *bitstream* created from another *bitstream* by operation of the *sub-bitstream extraction process* with the another *bitstream*, the target highest TemporalId equal to 6, and the target *layer identifier list* equal to the *layer identifier list* associated with the layer set as inputs.

3.78 **leading picture**: A *picture* that precedes the *associated IRAP picture* in *output order*.

3.79 **leaf**: A terminating node of a tree that is a root node of a tree of depth 0.

3.80 **level**: A defined set of constraints on the values that may be taken by the *syntax elements* and variables of this Specification, or the value of a *transform coefficient* prior to *scaling*.

NOTE – The same set of levels is defined for all profiles, with most aspects of the definition of each level being in common across different profiles. Individual implementations may, within the specified constraints, support a different level for each supported profile.

3.81 **list 0 (list 1) motion vector**: A *motion vector* associated with a *reference index* pointing into *reference picture list 0* (*list 1*).

3.82 **list 0 (list 1) prediction**: *Inter prediction* of the content of a *slice* using a *reference index* pointing into *reference picture list 0* (*list 1*).

3.83 **local coordinate axes**: The coordinate axes having a specified rotation relationship relative to the *global coordinate axes*.

3.84 **long-term reference picture**: A *picture* that is marked as "used for long-term reference".

3.85 **long-term reference picture set**: The two reference picture set (RPS) lists that may contain long-term reference pictures.

3.86 **luma**: An adjective, represented by the symbol or subscript Y or L, specifying that a sample array or single sample is representing the monochrome signal related to the primary colours.

NOTE – The term luma is used rather than the term luminance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term luminance. The symbol L is sometimes used instead of the symbol Y to avoid confusion with the symbol y as used for vertical location.

3.87 **may**: A term that is used to refer to behaviour that is allowed, but not necessarily required.

NOTE – In some places where the optional nature of the described behaviour is intended to be emphasized, the phrase "may or may not" is used to provide emphasis.

3.88 **motion vector**: A two-dimensional vector used for *inter prediction* that provides an offset from the coordinates in the *decoded picture* to the coordinates in a *reference picture*.

3.89 **must**: A term that is used in expressing an observation about a requirement or an implication of a requirement that is specified elsewhere in this Specification (used exclusively in an *informative* context).

3.90 **network abstraction layer (NAL) unit**: A *syntax structure* containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.

3.91 **network abstraction layer (NAL) unit stream**: A sequence of *NAL units*.

3.92 **non-scalable-nested SEI message**: An SEI message that is not contained in a scalable nesting SEI message.

3.93 **non-reference picture**: A *picture* that is marked as "unused for reference".

NOTE – A non-reference picture contains samples that cannot be used for inter prediction in the decoding process of subsequent pictures in decoding order. In other words, once a picture is marked as "unused for reference", it can never be marked back as "used for reference".

3.94 **non-VCL NAL unit**: A *NAL unit* that is not a *VCL NAL unit*.

3.95 **note**: A term that is used to prefix *informative* remarks (used exclusively in an *informative* context).

3.96 **omnidirectional video**: A video content in a format that enables rendering according to the user's viewing orientation, e.g., if viewed using a head-mounted device, or according to a user's desired *viewport*, reflecting a potentially rotated viewing position.

3.97 **operation point**: A *bitstream* created from another *bitstream* by operation of the *sub-bitstream extraction process* with the another *bitstream*, a target highest TemporalId, and a target *layer identifier list* as inputs.

NOTE – If the target highest TemporalId of an operation point is equal to the greatest value of TemporalId in the layer set associated with the target layer identification list, the operation point is identical to the layer set. Otherwise it is a subset of the layer set.

3.98 **output order**: The order in which the *decoded pictures* are output from the *decoded picture buffer* (for the *decoded pictures* that are to be output from the *decoded picture buffer*).

3.99 **output time**: A time when a *decoded picture* is to be output as specified by the *hypothetical reference decoder (HRD)* according to the output timing *decoded picture buffer (DPB)* operation.

3.100 **packed region**: A region in a *region-wise packed picture* that is mapped to a *projected region* according to a *region-wise packing*.

3.101 **parameter**: A *syntax element* of a *video parameter set (VPS), sequence parameter set (SPS)* or *picture parameter set (PPS)*, or the second word of the defined term *quantization parameter*.

3.102 **partitioning**: The division of a set into subsets such that each element of the set is in exactly one of the subsets.

3.103 **picture**: An array of *luma* samples in monochrome format or an array of *luma* samples and two corresponding arrays of *chroma* samples in 4:2:0, 4:2:2, and 4:4:4 colour format.

NOTE – A picture may be either a frame or a field. However, in one CVS, either all pictures are frames or all pictures are fields.

3.104 **picture parameter set (PPS)**: A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded pictures* as determined by a *syntax element* found in each *slice segment header*.

3.105 **picture order count (POC)**: A variable that is associated with each *picture*, uniquely identifies the associated *picture* among all *pictures* in the *CVS*, and, when the associated *picture* is to be output from the *decoded picture buffer*, indicates the position of the associated *picture* in *output order* relative to the *output order* positions of the other *pictures* in the same *CVS* that are to be output from the *decoded picture buffer*.

3.106 **picture unit**: A set of *NAL units* that contain all *VCL NAL units* of a *coded picture* and their *associated non-VCL NAL units*.

3.107 **prediction**: An embodiment of the *prediction process*.

3.108 **prediction block**: A rectangular MxN *block* of samples on which the same *prediction* is applied.

3.109 **prediction process**: The use of a *predictor* to provide an estimate of the data element (e.g., sample value or motion vector) currently being decoded.

3.110 **prediction unit**: A *prediction block* of *luma* samples, two corresponding *prediction blocks* of *chroma* samples of a *picture* that has three sample arrays, or a *prediction block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes and *syntax structures* used to predict the *prediction block* samples.

3.111 **predictive (P) slice**: A *slice* that is decoded using *intra prediction* or using *inter prediction* with at most one *motion vector* and *reference index* to *predict* the sample values of each *block*.

3.112 **predictor**: A combination of specified values or previously decoded data elements (e.g., sample value or motion vector) used in the *decoding process* of subsequent data elements.

3.113 **prefix SEI message**: An SEI message that is contained in a *prefix SEI NAL unit*.

3.114 **prefix SEI NAL unit**: An *SEI NAL unit* that has nal_unit_type equal to PREFIX_SEI_NUT.

3.115 **profile**: A specified subset of the syntax of this Specification.

3.116 **projected picture**: A picture that uses a *projection* format for *omnidirectional video*.

3.117 **projected region**: A region in a *projected picture* that is mapped to a *packed region* according to a *region-wise packing*.

3.118 **projection**: A specified correspondence between the colour samples of a *projected picture* and azimuth and elevation positions on a sphere.

3.119 **pulse code modulation (PCM)**: Coding of the samples of a *block* by directly representing the sample values without *prediction* or application of a transform.

3.120 **quadtree**: A *tree* in which a parent node can be split into four child nodes, each of which may become parent node for another split into four child nodes.

3.121 **quantization parameter**: A variable used by the *decoding process* for *scaling* of *transform coefficient levels*.

3.122 **random access**: The act of starting the decoding process for a *bitstream* at a point other than the beginning of the stream.

3.123 **random access decodable leading (RADL) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is a *RADL picture*.

3.124 **random access decodable leading (RADL) picture**: A *coded picture* for which each *VCL NAL unit* has nal_unit_type equal to RADL_R or RADL_N.

NOTE – All RADL pictures are leading pictures. RADL pictures are not used as reference pictures for the decoding process of trailing pictures of the same associated IRAP picture. When present, all RADL pictures precede, in decoding order, all trailing pictures of the same associated IRAP picture.

3.125 **random access skipped leading (RASL) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is a *RASL picture*.

3.126 **random access skipped leading (RASL) picture**: A *coded picture* for which each *VCL NAL unit* has nal_unit_type equal to RASL_R or RASL_N.

NOTE – All RASL pictures are leading pictures of an associated BLA or CRA picture. When the associated IRAP picture has NoRaslOutputFlag equal to 1, the RASL picture is not output and may not be correctly decodable, as the RASL picture may contain references to pictures that are not present in the bitstream. RASL pictures are not used as reference pictures for the decoding process of non-RASL pictures. When present, all RASL pictures precede, in decoding order, all trailing pictures of the same associated IRAP picture.

3.127 **raster scan**: A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc., rows of the pattern (going down) each scanned from left to right.

3.128 **raw byte sequence payload (RBSP)**: A *syntax structure* containing an integer number of *bytes* that is encapsulated in a *NAL unit* and that is either empty or has the form of a *string of data bits* containing *syntax elements* followed by an *RBSP stop bit* and zero or more subsequent bits equal to 0.

3.129 **raw byte sequence payload (RBSP) stop bit**: A bit equal to 1 present within a *raw byte sequence payload (RBSP)* after a *string of data bits*, for which the location of the end within an *RBSP* can be identified by searching from the end of the *RBSP* for the *RBSP stop bit*, which is the last non-zero bit in the *RBSP*.

3.130 **recovery point**: A point in the *bitstream* at which the recovery of an exact or an approximate representation of the *decoded pictures* represented by the *bitstream* is achieved after a *random access* or *broken link*.

3.131 **reference index**: An index into a *reference picture list*.

3.132 **reference picture**: A *picture* that is a *short-term reference picture* or a *long-term reference picture*.

NOTE – A reference picture contains samples that may be used for inter prediction in the decoding process of subsequent pictures in decoding order.

3.133 **reference picture list**: A list of *reference pictures* that is used for *inter prediction* of a *P* or *B slice*.

NOTE – For the decoding process of a P slice, there is one reference picture list – reference picture list 0. For the decoding process of a B slice, there are two reference picture lists – reference picture list 0 and reference picture list 1.

3.134 **reference picture list 0**: The *reference picture list* used for *inter prediction* of a *P* or the first *reference picture list* used for *inter prediction* of a *B slice*.

3.135 **reference picture list 1**: The second *reference picture list* used for *inter prediction* of a *B slice*.

3.136 **reference picture set (RPS)**: A set of *reference pictures* associated with a *picture*, consisting of all *reference pictures* that are prior to the associated *picture* in *decoding order*, that may be used for *inter prediction* of the associated *picture* or any *picture* following the associated *picture* in *decoding order*.

NOTE – The RPS of a picture consists of five RPS lists, three of which are to contain short-term reference pictures and the other two are to contain long-term reference pictures.

3.137 **region-wise packed picture**: A decoded picture that contains one or more *packed regions*.

NOTE – A region-wise packed picture may contain a *region-wise packing* of a *projected picture*.

3.138 **region-wise packing**: A transformation, resizing, and relocation of *packed regions* of a *region-wise packed picture* to remap the *packed regions* to *projected region*s of a *projected picture*.

3.139 **reserved**: A term that may be used to specify that some values of a particular *syntax element* are for future use by ITU-T | ISO/IEC and shall not be used in *bitstreams* conforming to this version of this Specification, but may be used in bitstreams conforming to future extensions of this Specification by ITU-T | ISO/IEC.

3.140 **residual**: The decoded difference between a *prediction* of a sample or data element and its decoded value.

3.141 **sample aspect ratio**: The ratio between the intended horizontal distance between the columns and the intended vertical distance between the rows of the *luma* sample array in a *picture*, which is specified for assisting the *display process* (not specified in this Specification) and expressed as *h*:*v*, where *h* is the horizontal width and *v* is the vertical height, in arbitrary units of spatial distance.

3.142 **scalable-nested SEI message**: An SEI message that is contained in a scalable nesting SEI message.

3.143 **scaling**: The process of multiplying *transform coefficient levels* by a factor, resulting in *transform coefficients*.

3.144 **sequence parameter set (SPS)**: A *syntax structure* containing *syntax elements* that apply to zero or more entire *CVSs* as determined by the content of a *syntax element* found in the *PPS* referred to by a *syntax element* found in each *slice segment header*.

3.145 **shall**: A term used to express mandatory requirements for conformance to this Specification.

NOTE – When used to express a mandatory constraint on the values of syntax elements or on the results obtained by operation of the specified decoding process, it is the responsibility of the encoder to ensure that the constraint is fulfilled. When used in reference to operations performed by the decoding process, any decoding process that produces identical cropped decoded pictures to those output from the decoding process described in this Specification conforms to the decoding process requirements of this Specification.

3.146 **short-term reference picture**: A *picture* that is marked as "used for short-term reference".

3.147 **short-term reference picture set**: The three RPS lists that may contain short-term reference pictures.

3.148 **should**: A term used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Specification.

3.149 **slice**: An integer number of *CTUs* contained in one *independent slice segment* and all subsequent *dependent slice segments* (if any) that precede the next *independent slice segment* (if any) within the same *access unit*.

3.150 **slice header**: The *slice segment header* of the *independent slice segment* that is a current *slice segment* or the most recent *independent slice segment* that precedes a current *dependent slice segment* in *decoding order*.

3.151 **slice segment**: An integer number of *CTUs* ordered consecutively in the *tile scan* and contained in a single *NAL unit*.

3.152 **slice segment header**: A part of a coded *slice segment* containing the data elements pertaining to the first or all *CTUs* represented in the *slice segment*.

3.153 **source**: A term used to describe the video material or some of its attributes before encoding.

3.154 **sphere coordinates**: The azimuth and elevation angles identifying a location of a point on a sphere.

3.155 **sphere region**: A region on a sphere, specified either by four *great circles* or by two *azimuth circles* and two *elevation circles*, or such a region on a rotated sphere after applying yaw, pitch, and roll rotations.

3.156 **start code prefix**: A unique sequence of three *bytes* equal to 0x000001 embedded in the *byte stream* as a prefix to each *NAL unit*.

NOTE – The location of a start code prefix can be used by a decoder to identify the beginning of a new NAL unit and the end of a previous NAL unit. Emulation of start code prefixes is prevented within NAL units by the inclusion of emulation prevention bytes.

3.157 **step-wise temporal sub-layer access (STSA) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is an *STSA picture*.

3.158 **step-wise temporal sub-layer access (STSA) picture**: A *coded picture* for which each *VCL NAL unit* has nal_unit_type equal to STSA_R or STSA_N.

NOTE – An STSA picture does not use pictures with the same TemporalId as the STSA picture for inter prediction reference. Pictures following an STSA picture in decoding order with the same TemporalId as the STSA picture do not use pictures prior to the STSA picture in decoding order with the same TemporalId as the STSA picture for inter prediction reference. An STSA picture enables up-switching, at the STSA picture, to the sub-layer containing the STSA picture, from the immediately lower sub-layer. STSA pictures must have TemporalId greater than 0.

3.159 **string of data bits (SODB)**: A sequence of some number of bits representing *syntax elements* present within a *raw byte sequence payload* prior to the *raw byte sequence payload stop bit*, where the left-most bit is considered to be the first and most significant bit, and the right-most bit is considered to be the last and least significant bit.

3.160 **sub-bitstream extraction process**: A specified process by which *NAL units* in a *bitstream* that do not belong to a target set, determined by a target highest TemporalId and a target *layer identifier list*, are removed from the *bitstream*, with the output sub-bitstream consisting of the NAL units in the *bitstream* that belong to the target set.

3.161 **sub-layer**: A temporal scalable layer of a temporal scalable *bitstream*, consisting of *VCL NAL units* with a particular value of the TemporalId variable and the associated *non-VCL NAL units*.

3.162 **sub-layer non-reference (SLNR) picture**: A *picture* that contains samples that cannot be used for *inter prediction* in the *decoding process* of subsequent *pictures* of the same *sub-layer* in *decoding order*.

NOTE – Samples of an SLNR picture may be used for inter prediction in the decoding process of subsequent pictures of higher sub-layers in decoding order.

3.163 **sub-layer reference picture**: A *picture* that contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* of the same *sub-layer* in *decoding order*.

NOTE – Samples of a sub-layer reference picture may also be used for inter prediction in the decoding process of subsequent pictures of higher sub-layers in decoding order.

3.164 **sub-layer representation**: A subset of the *bitstream* consisting of *NAL units* of a particular *sub-layer* and the lower *sub-layers*.

3.165 **suffix SEI message**: An SEI message that is contained in a *suffix SEI NAL unit*.

3.166 **suffix SEI NAL unit**: An *SEI NAL unit* that has nal_unit_type equal to SUFFIX_SEI_NUT.

3.167 **supplemental enhancement information (SEI) NAL unit**: A *NAL unit* that has nal_unit_type equal to PREFIX_SEI_NUT or SUFFIX_SEI_NUT.

3.168 **syntax element**: An element of data represented in the *bitstream*.

3.169 **syntax structure**: Zero or more *syntax elements* present together in the *bitstream* in a specified order.

3.170 **temporal sub-layer**: Same as *sub-layer*.

3.171 **temporal sub-layer access (TSA) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to 0 is a *TSA picture*.

3.172 **temporal sub-layer access (TSA) picture**: A *coded picture* for which each *VCL NAL unit* has nal_unit_type equal to TSA_R or TSA_N.

NOTE – A TSA picture and pictures following the TSA picture in decoding order do not use pictures prior to the TSA picture in decoding order with TemporalId greater than or equal to that of the TSA picture for inter prediction reference. A TSA picture enables up-switching, at the TSA picture, to the sub-layer containing the TSA picture or any higher sub-layer, from the immediately lower sub-layer. TSA pictures must have TemporalId greater than 0.

3.173 **tier**: A specified category of *level* constraints imposed on values of the *syntax elements* in the *bitstream*, where the *level* constraints are nested within a *tier* and a *decoder* conforming to a certain *tier* and *level* would be capable of decoding all *bitstreams* that conform to the same *tier* or the lower *tier* of that *level* or any *level* below it.

3.174 **tile**: A rectangular region of *CTUs* within a particular *tile column* and a particular *tile row* in a *picture*.

3.175 **tile column**: A rectangular region of *CTUs* having a height equal to the height of the *picture* and a width specified by *syntax elements* in the *picture parameter set*.

3.176 **tile row**: A rectangular region of *CTUs* having a height specified by *syntax elements* in the *picture parameter set* and a width equal to the width of the *picture*.

3.177 **tile scan**: A specific sequential ordering of *CTUs partitioning* a *picture* in which the *CTUs* are ordered consecutively in *CTU raster scan* in a *tile*, whereas *tiles* in a *picture* are ordered consecutively in a *raster scan* of the *tiles* of the *picture*.

3.178 **tilt angle**: The angle indicating the amount of tilt of a *sphere region*, measured as the amount of rotation of a *sphere region* along the axis originating from the sphere origin passing through the centre point of the *sphere region*, where the angle value increases clockwise when looking from the origin towards the positive end of the axis.

3.179 **trailing picture**: A non-IRAP *picture* that follows the *associated IRAP picture* in *output order*.

NOTE – Trailing pictures associated with an IRAP picture also follow the IRAP picture in decoding order. Pictures that follow the associated IRAP picture in output order and precede the associated IRAP picture in decoding order are not allowed.

3.180 **transform**: A part of the *decoding process* by which a *block* of *transform coefficients* is converted to a *block* of spatial-domain values.

3.181 **transform block**: A rectangular MxN *block* of samples resulting from a *transform* in the *decoding process*.

3.182 **transform coefficient**: A scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* in a *transform* in the *decoding process*.

3.183 **transform coefficient level**: An integer quantity representing the value associated with a particular two-dimensional frequency index in the *decoding process* prior to *scaling* for computation of a *transform coefficient* value.

3.184 **transform unit**: A *transform block* of *luma* samples of size 8x8, 16x16, or 32x32 or four *transform blocks* of *luma* samples of size 4x4, two corresponding *transform blocks* of *chroma* samples of a *picture* in 4:2:0 colour format; or a *transform block* of *luma* samples of size 8x8, 16x16, or 32x32, and four corresponding *transform blocks* of *chroma* samples, or four *transform blocks* of *luma* samples of size 4x4, and four corresponding *transform blocks* of *chroma* samples of a *picture* in 4:2:2 colour format; or a *transform block* of *luma* samples of size 4x4, 8x8, 16x16, or 32x32, and two corresponding *transform blocks* of *chroma* samples of a *picture* in 4:4:4 colour format that is not coded using three separate colour planes and *syntax structures* used to transform the *transform block* samples; or a *transform block* of *luma* samples of size 8x8, 16x16, or 32x32 or four *transform blocks* of *luma* samples of size 4x4 of a monochrome *picture* or a *picture* in 4:4:4 colour format that is coded using three separate colour planes; and the associated *syntax structures* used to transform the *transform block* samples.

3.185 **tree**: A tree is a finite set of nodes with a unique root node.

3.186 **universal unique identifier (UUID)**: An identifier that is unique with respect to the space of all universal unique identifiers.

3.187 **unspecified**: A term that may be used to specify some values of a particular *syntax element* to indicate that the values have no specified meaning in this Specification and will not have a specified meaning in the future as an integral part of future versions of this Specification.

3.188 **video coding layer (VCL) NAL unit**: A collective term for *coded slice segment NAL units* and the subset of *NAL units* that have *reserved* values of nal_unit_type that are classified as VCL NAL units in this Specification.

3.189 **video parameter set (VPS)**: A *syntax structure* containing *syntax elements* that apply to zero or more entire *CVSs* as determined by the content of a *syntax element* found in the *SPS* referred to by a *syntax element* found in the *PPS* referred to by a *syntax element* found in each *slice segment header*.

3.190 **viewport**: A region of *omnidirectional video* content suitable for display and viewing by the user.

3.191 **z-scan order**: A specified sequential ordering of *blocks partitioning* a *picture*, where the order is identical to *CTB raster scan* of the *picture* when the *blocks* are of the same size as *CTBs*, and, when the *blocks* are of a smaller size than *CTBs*, i.e., *CTBs* are further partitioned into smaller *coding blocks*, the order traverses from *CTB* to *CTB* in *CTB raster scan* of the *picture*, and inside each *CTB*, which may be divided into *quadtrees* hierarchically to lower levels, the order traverses from *quadtree* to *quadtree* of a particular level in *quadtree*-of-the-particular-level *raster scan* of the *quadtree* of the immediately higher level.

4 Abbreviations and acronyms

For the purposes of this Recommendation | International Standard, the following abbreviations and acronyms apply:

ATSC Advanced Television Systems Committee

B Bi-predictive

BLA Broken Link Access

BPB	Bitstream Partition Buffer
CABAC	Context-based Adaptive Binary Arithmetic Coding
CB	Coding Block
CBR	Constant Bit Rate
CIE	International Commission on Illumination (Commission Internationale de l'Eclairage)
CLVS	Coded Layer-wise Video Sequence
CPB	Coded Picture Buffer
CRA	Clean Random Access
CRC	Cyclic Redundancy Check
CTB	Coding Tree Block
CTU	Coding Tree Unit
CU	Coding Unit
CVS	Coded Video Sequence
CVSG	Coded Video Sequence Group
DCT	Discrete Cosine Transform
DPB	Decoded Picture Buffer
DRAP	Dependent Random Access Point
DUT	Decoder Under Test
EG	Exponential-Golomb
EGk	k-th order Exponential-Golomb
FCC	Federal Communications Commission (of the United States)
FIFO	First-In, First-Out
FIR	Finite Impulse Response
FL	Fixed-Length
GBR	Green, Blue and Red
GDR	Gradual Decoding Refresh
HRD	Hypothetical Reference Decoder
HSS	Hypothetical Stream Scheduler
I	Intra
IDCT	Inverse Discrete Cosine Transformation
IDR	Instantaneous Decoding Refresh
INBLD	Independent Non-Base Layer Decoding
IRAP	Intra Random Access Point
LPS	Least Probable Symbol
LSB	Least Significant Bit
MCTS	Motion-Constrained Tile Set
MAC	Multiplexed Analogue Components
MPS	Most Probable Symbol
MSB	Most Significant Bit
MVP	Motion Vector Prediction

NAL	Network Abstraction Layer
NTSC	National Television System Committee (of the United States)
OLS	Output Layer Set
P	Predictive
PAL	Phase Alternating Line
PB	Prediction Block
PCM	Pulse Code Modulation
POC	Picture Order Count
PPS	Picture Parameter Set
PU	Prediction Unit
QP	Quantization Parameter
RADL	Random Access Decodable Leading (Picture)
RASL	Random Access Skipped Leading (Picture)
RBSP	Raw Byte Sequence Payload
RGB	Same as GBR
RPS	Reference Picture Set
SAO	Sample Adaptive Offset
SAR	Sample Aspect Ratio
SECAM	Sequential colour with memory (Séquentiel Couleur avec Mémoire)
SEI	Supplemental Enhancement Information
SLNR	Sub-Layer Non-Reference (Picture)
SMPTE	Society of Motion Picture and Television Engineers
SODB	String Of Data Bits
SPS	Sequence Parameter Set
STSA	Step-wise Temporal Sub-layer Access
TB	Transform Block
TR	Truncated Rice
TSA	Temporal Sub-layer Access
TU	Transform Unit
UCS	Universal Coded Character Set
UTF	UCS Transmission Format
UUID	Universal Unique Identifier
VBR	Variable Bit Rate
VCL	Video Coding Layer
VPS	Video Parameter Set
VUI	Video Usability Information

5 Conventions

5.1 General

NOTE – The mathematical operators used in this Specification are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division. Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

5.2 Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
−	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, 7 / 4 and −7 / −4 are truncated to 1 and −7 / 4 and 7 / −4 are truncated to −1.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^{y} f(i)$	The summation of f(i) with i taking all integer values from x up to and including y.
x % y	Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0.

5.3 Logical operators

The following logical operators are defined as follows:

x && y	Boolean logical "and" of x and y
x || y	Boolean logical "or" of x and y
!	Boolean logical "not"
x ? y : z	If x is TRUE or not equal to 0, evaluates to the value of y; otherwise, evaluates to the value of z.

5.4 Relational operators

The following relational operators are defined as follows:

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
= =	Equal to
!=	Not equal to

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

5.5 Bit-wise operators

The following bit-wise operators are defined as follows:

&	Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

| Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

^ Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

x >> y Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the most significant bits (MSBs) as a result of the right shift have a value equal to the MSB of x prior to the shift operation.

x << y Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the least significant bits (LSBs) as a result of the left shift have a value equal to 0.

5.6 Assignment operators

The following arithmetic operators are defined as follows:

= Assignment operator

++ Increment, i.e., *x*++ is equivalent to *x* = *x* + 1; when used in an array index, evaluates to the value of the variable prior to the increment operation.

−− Decrement, i.e., *x*−− is equivalent to *x* = *x* − 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation.

+= Increment by amount specified, i.e., x += 3 is equivalent to x = x + 3, and x += (−3) is equivalent to x = x + (−3).

−= Decrement by amount specified, i.e., x −= 3 is equivalent to x = x − 3, and x −= (−3) is equivalent to x = x − (−3).

5.7 Range notation

The following notation is used to specify a range of values:

x = y..z x takes on integer values starting from y to z, inclusive, with x, y, and z being integer numbers and z being greater than or equal to y.

5.8 Mathematical functions

The following mathematical functions are defined:

$$\text{Abs}(\ x\ ) = \begin{cases} x & ;\quad x >= 0 \\ -x & ;\quad x < 0 \end{cases} \tag{5-1}$$

Asin(x) the trigonometric inverse sine function, operating on an argument x that is in the range of −1.0 to 1.0, inclusive, with an output value in the range of −π÷2 to π÷2, inclusive, in units of radians (5-2)

Atan(x) the trigonometric inverse tangent function, operating on an argument x, with an output value in the range of −π÷2 to π÷2, inclusive, in units of radians (5-3)

$$\text{Atan2}(\ y, x\ ) = \begin{cases} \text{Atan}\left(\frac{y}{x}\right) & ;\quad x > 0 \\ \text{Atan}\left(\frac{y}{x}\right) + \pi & ;\quad x < 0\ \&\&\ y >= 0 \\ \text{Atan}\left(\frac{y}{x}\right) - \pi & ;\quad x < 0\ \&\&\ y < 0 \\ +\frac{\pi}{2} & ;\quad x == 0\ \&\&\ y >= 0 \\ -\frac{\pi}{2} & ;\quad \text{otherwise} \end{cases} \tag{5-4}$$

Ceil(x) the smallest integer greater than or equal to x. (5-5)

$$\text{Clip1}_Y(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_Y) - 1, x) \quad (5\text{-}6)$$

$$\text{Clip1}_C(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_C) - 1, x) \quad (5\text{-}7)$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases} \quad (5\text{-}8)$$

Cos(x) the trigonometric cosine function operating on an argument x in units of radians. (5-9)

Floor(x) the largest integer less than or equal to x. (5-10)

$$\text{GetCurrMsb}(a, b, c, d) = \begin{cases} c + d & ; \quad b - a >= d / 2 \\ c - d & ; \quad a - b > d / 2 \\ c & ; \quad \text{otherwise} \end{cases} \quad (5\text{-}11)$$

Ln(x) the natural logarithm of x (the base-e logarithm, where e is the natural logarithm base constant 2.718 281 828...). (5-12)

Log2(x) the base-2 logarithm of x. (5-13)

Log10(x) the base-10 logarithm of x. (5-14)

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x <= y \\ y & ; \quad x > y \end{cases} \quad (5\text{-}15)$$

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x >= y \\ y & ; \quad x < y \end{cases} \quad (5\text{-}16)$$

$$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0.5) \quad (5\text{-}17)$$

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x > 0 \\ 0 & ; \quad x == 0 \\ -1 & ; \quad x < 0 \end{cases} \quad (5\text{-}18)$$

Sin(x) the trigonometric sine function operating on an argument x in units of radians (5-19)

Sqrt(x) the square root of x (5-20)

Swap(x, y) = (y, x) (5-21)

Tan(x) the trigonometric tangent function operating on an argument x in units of radians (5-22)

5.9 Order of operation precedence

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

– Operations of a higher precedence are evaluated before any operation of a lower precedence.

– Operations of the same precedence are evaluated sequentially from left to right.

Table 5-1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x− −"
"!x", "−x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", "$\frac{x}{y}$", "x % y"
"x + y", "x − y" (as a two-argument operator), " $\sum_{i=x}^{y} f(i)$ "
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x = = y", "x != y"
"x & y"
"x | y"
"x && y"
"x || y"
"x ? y : z"
"x..y"
"x = y", "x += y", "x −= y"

5.10 Variables, syntax elements and tables

Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), and one descriptor for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letters and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the clause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions that specify properties of the current position in the bitstream are referred to as syntax functions. These functions are specified in clause 7.2 and assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream. Syntax functions are described by their names, which are constructed as syntax element names and end with left and right round parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

Functions that are not syntax functions (including mathematical functions specified in clause 5.8) are described by their names, which start with an upper case letter, contain a mixture of lower and upper case letters without any underscore character, and end with left and right parentheses including zero or more variable names (for definition) or values (for usage) separated by commas (if more than one variable).

A one-dimensional array is referred to as a list. A two-dimensional array is referred to as a matrix. Arrays can either be syntax elements or variables. Subscripts or square parentheses are used for the indexing of arrays. In reference to a visual depiction of a matrix, the first subscript is used as a row (vertical) index and the second subscript is used as a column (horizontal) index. The indexing order is reversed when using square parentheses rather than subscripts for indexing.

Thus, an element of a matrix s at horizontal position x and vertical position y may be denoted either as s[x][y] or as s_{yx}. A single column of a matrix may be referred to as a list and denoted by omission of the row index. Thus, the column of a matrix s at horizontal position x may be referred to as the list s[x].

A specification of values of the entries in rows and columns of an array may be denoted by { {...} {...} }, where each inner pair of brackets specifies the values of the elements within a row in increasing column order and the rows are ordered in increasing row order. Thus, setting a matrix s equal to { { 1 6 } { 4 9 }} specifies that s[0][0] is set equal to 1, s[1][0] is set equal to 6, s[0][1] is set equal to 4, and s[1][1] is set equal to 9.

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any value different from zero.

5.11 Text description of logical operations

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0 )
  statement 0
else if( condition 1 )
  statement 1
...
else /* informative remark on remaining condition */
  statement n
```

may be described in the following manner:

... as follows / ... the following applies:

– If condition 0, statement 0
– Otherwise, if condition 1, statement 1
– ...
– Otherwise (informative remark on remaining condition), statement n

Each "If ... Otherwise, if ... Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If ... Otherwise, if ... Otherwise, ..." is always an "Otherwise, ...". Interleaved "If ... Otherwise, if ... Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0a && condition 0b )
  statement 0
else if( condition 1a || condition 1b )
  statement 1
...
else
  statement n
```

may be described in the following manner:

... as follows / ... the following applies:

– If all of the following conditions are true, statement 0:
  – condition 0a
  – condition 0b
– Otherwise, if one or more of the following conditions are true, statement 1:

    – condition 1a
    – condition 1b
  – ...
  – Otherwise, statement n

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0 )
  statement 0
if( condition 1 )
  statement 1
```

may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1

5.12 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows:

– If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.
– Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

In the specification of a process, a specific coding block may be referred to by the variable name having a value equal to the address of the specific coding block.

6 Bitstream and picture formats, partitionings, scanning processes and neighbouring relationships

6.1 Bitstream formats

This clause specifies the relationship between the network abstraction layer (NAL) unit stream and byte stream, either of which are referred to as the bitstream.

The bitstream can be in one of two formats: the NAL unit stream format or the byte stream format. The NAL unit stream format is conceptually the more "basic" type. It consists of a sequence of syntax structures called NAL units. This sequence is ordered in decoding order. There are constraints imposed on the decoding order (and contents) of the NAL units in the NAL unit stream.

The byte stream format can be constructed from the NAL unit stream format by ordering the NAL units in decoding order and prefixing each NAL unit with a start code prefix and zero or more zero-valued bytes to form a stream of bytes. The NAL unit stream format can be extracted from the byte stream format by searching for the location of the unique start code prefix pattern within this stream of bytes. Methods of framing the NAL units in a manner other than use of the byte stream format are outside the scope of this Specification. The byte stream format is specified in Annex B.

6.2 Source, decoded and output picture formats

This clause specifies the relationship between source and decoded pictures that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of pictures in decoding order.

The source and decoded pictures are each comprised of one or more sample arrays:

– Luma (Y) only (monochrome).
– Luma and two chroma (YCbCr or YCgCo).

– Green, blue, and red (GBR, also known as RGB).

– Arrays representing other unspecified monochrome or tri-stimulus colour samplings (for example, YZX, also known as XYZ).

For convenience of notation and terminology in this Specification, the variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma, where the two chroma arrays are referred to as Cb and Cr; regardless of the actual colour representation method in use. The actual colour representation method in use can be indicated in syntax that is specified in Annex E.

The variables SubWidthC and SubHeightC are specified in Table 6-1, depending on the chroma format sampling structure, which is specified through chroma_format_idc and separate_colour_plane_flag. Other values of chroma_format_idc, SubWidthC and SubHeightC may be specified in the future by ITU-T | ISO/IEC.

Table 6-1 – SubWidthC and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag

chroma_format_idc	separate_colour_plane_flag	Chroma format	SubWidthC	SubHeightC
0	0	Monochrome	1	1
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	1	1

In monochrome sampling there is only one sample array, which is nominally considered the luma array.

In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

In 4:2:2 sampling, each of the two chroma arrays has the same height and half the width of the luma array.

In 4:4:4 sampling, depending on the value of separate_colour_plane_flag, the following applies:

– If separate_colour_plane_flag is equal to 0, each of the two chroma arrays has the same height and width as the luma array.

– Otherwise (separate_colour_plane_flag is equal to 1), the three colour planes are separately processed as monochrome sampled pictures.

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 16, inclusive, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

When the value of chroma_format_idc is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in pictures are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

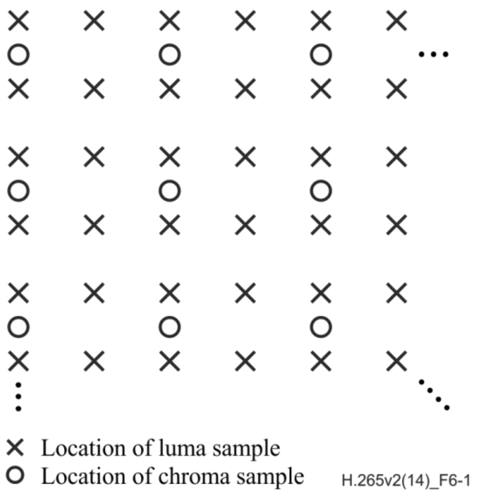


Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a picture

When the value of chroma_format_idc is equal to 2, the chroma samples are co-sited with the corresponding luma samples and the nominal locations in a picture are as shown in Figure 6-2.

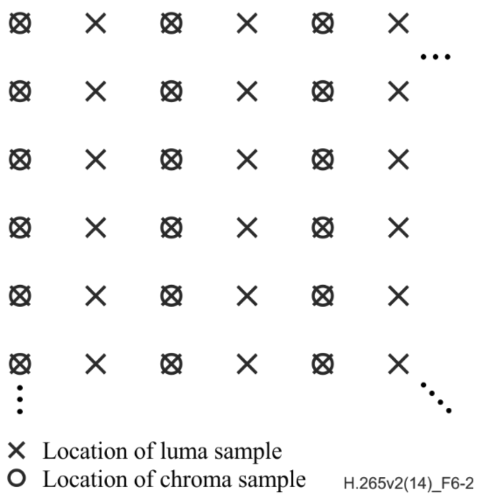


Figure 6-2 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture

When the value of chroma_format_idc is equal to 3, all array samples are co-sited for all cases of pictures and the nominal locations in a picture are as shown in Figure 6-3.

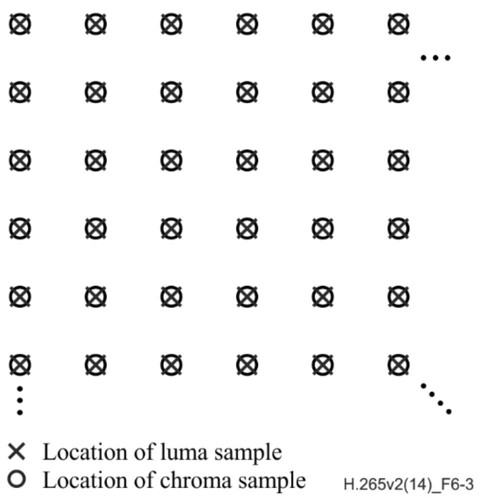


Figure 6-3 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture

6.3 Partitioning of pictures, slices, slice segments, tiles, CTUs and CTBs

6.3.1 Partitioning of pictures into slices, slice segments and tiles

This clause specifies how a picture is partitioned into slices, slice segments and tiles. Pictures are divided into slices and tiles. A slice is a sequence of one or more slice segments starting with an independent slice segment and containing all subsequent dependent slice segments (if any) that precede the next independent slice segment (if any) within the same picture. A slice segment is a sequence of CTUs. Likewise, a tile is a sequence of CTUs.

For example, a picture may be divided into two slices as shown in Figure 6-4. In this example, the first slice is composed of an independent slice segment containing 4 CTUs, a dependent slice segment containing 32 CTUs and another dependent slice segment containing 24 CTUs; and the second slice consists of a single independent slice segment containing the remaining 39 CTUs of the picture.

As another example, a picture may be divided into two tiles separated by a vertical tile boundary as shown in Figure 6-5. The left side of the figure illustrates a case in which the picture only contains one slice, starting with an independent slice segment and followed by four dependent slice segments. The right side of the figure illustrates an alternative case in which the picture contains two slices in the first tile and one slice in the second tile.

Unlike slices, tiles are always rectangular. A tile always contains an integer number of CTUs, and may consist of CTUs contained in more than one slice. Similarly, a slice may consist of CTUs contained in more than one tile.

One or both of the following conditions shall be fulfilled for each slice and tile:

– All CTUs in a slice belong to the same tile.

– All CTUs in a tile belong to the same slice.

NOTE 1 – Within the same picture, there may be both slices that contain multiple tiles and tiles that contain multiple slices.

One or both of the following conditions shall be fulfilled for each slice segment and tile:

– All CTUs in a slice segment belong to the same tile.

– All CTUs in a tile belong to the same slice segment.

When a picture is coded using three separate colour planes (separate_colour_plane_flag is equal to 1), a slice contains only CTBs of one colour component being identified by the corresponding value of colour_plane_id, and each colour component array of a picture consists of slices having the same colour_plane_id value. Coded slices with different values of colour_plane_id within a picture may be interleaved with each other under the constraint that for each value of colour_plane_id, the coded slice segment NAL units with that value of colour_plane_id shall be in the order of increasing CTB address in tile scan order for the first CTB of each coded slice segment NAL unit.

NOTE 2 – When separate_colour_plane_flag is equal to 0, each CTB of a picture is contained in exactly one slice. When separate_colour_plane_flag is equal to 1, each CTB of a colour component is contained in exactly one slice (i.e., information for each CTB of a picture is present in exactly three slices and these three slices have different values of colour_plane_id).

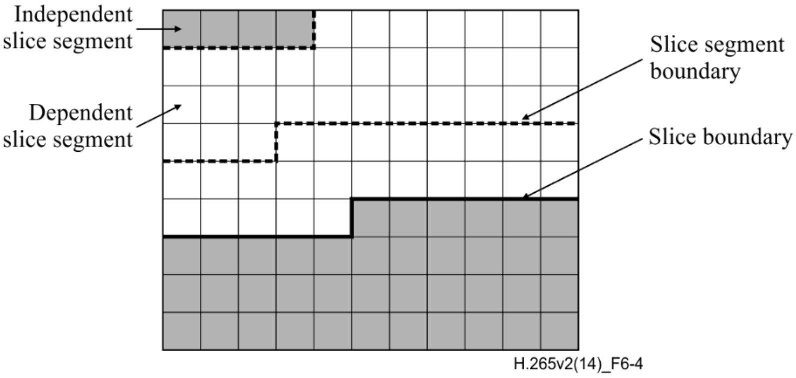


Figure 6-4 – A picture with 11 by 9 luma CTBs that is partitioned into two slices, the first of which is partitioned into three slice segments (informative)

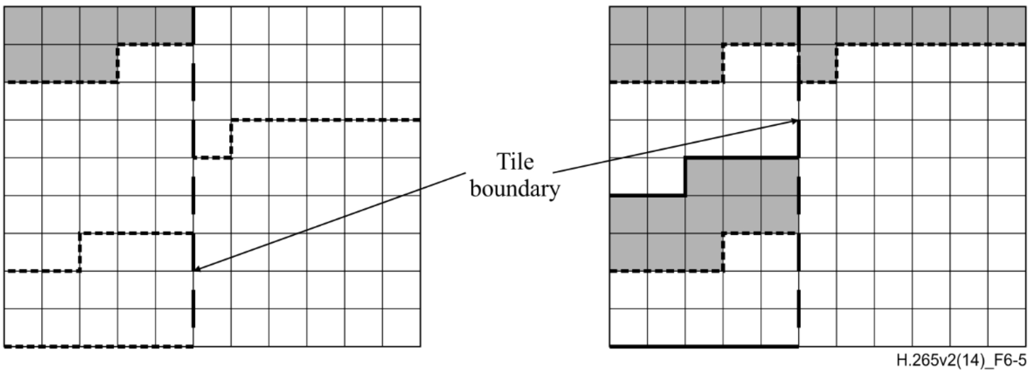


Figure 6-5 – A picture with 11 by 9 luma CTBs that is partitioned into two tiles and one slice (left) or is partitioned into two tiles and three slices (right) (informative)

6.3.2 Block and quadtree structures

The samples are processed in units of CTBs. The array size for each luma CTB in both width and height is CtbSizeY in units of samples. The width and height of the array for each chroma CTB are CtbWidthC and CtbHeightC, respectively, in units of samples.

Each CTB is assigned a partition signalling to identify the block sizes for intra or inter prediction and for transform coding. The partitioning is a recursive quadtree partitioning. The root of the quadtree is associated with the CTB. The quadtree is split until a leaf is reached, which is referred to as the coding block. When the component width is not an integer number of the CTB size, the CTBs at the right component boundary are incomplete. When the component height is not an integer multiple of the CTB size, the CTBs at the bottom component boundary are incomplete.

The coding block is the root node of two trees, the prediction tree and the transform tree. The prediction tree specifies the position and size of prediction blocks. The transform tree specifies the position and size of transform blocks. The splitting information for luma and chroma is identical for the prediction tree and may or may not be identical for the transform tree.

The blocks and associated syntax structures are grouped into "unit" structures as follows:

– One prediction block (monochrome picture or separate_colour_plane_flag is equal to 1) or three prediction blocks (luma and chroma components of a picture in 4:2:0 or 4:4:4 colour format) or five prediction blocks (luma and chroma components of a picture in 4:2:2 colour format) and the associated prediction syntax structures units are associated with a prediction unit.

– One transform block (monochrome picture or separate_colour_plane_flag is equal to 1) or three transform blocks (luma and chroma components of a picture in 4:2:0 or 4:4:4 colour format) or five transform blocks (luma and chroma components of a picture in 4:2:2 colour format) and the associated transform syntax structures units are associated with a transform unit.

– One coding block (monochrome picture or separate_colour_plane_flag is equal to 1) or three coding blocks (luma and chroma), the associated coding syntax structures and the associated prediction and transform units are associated with a coding unit.

– One CTB (monochrome picture or separate_colour_plane_flag is equal to 1) or three CTBs (luma and chroma), the associated coding tree syntax structures and the associated coding units are associated with a CTU.

6.3.3 Spatial or component-wise partitionings

The following divisions of processing elements of this Specification form spatial or component-wise partitionings:

– The division of each picture into components

– The division of each component into CTBs

– The division of each picture into tile columns

– The division of each picture into tile rows

– The division of each tile column into tiles

– The division of each tile row into tiles

– The division of each tile into CTUs

– The division of each picture into slices

– The division of each slice into slice segments

– The division of each slice segment into CTUs

– The division of each CTU into CTBs

– The division of each CTB into coding blocks, except that the CTBs are incomplete at the right component boundary when the component width is not an integer multiple of the CTB size and the CTBs are incomplete at the bottom component boundary when the component height is not an integer multiple of the CTB size

– The division of each CTU into coding units, except that the CTUs are incomplete at the right picture boundary when the picture width in luma samples is not an integer multiple of the luma CTB size and the CTUs are incomplete at the bottom picture boundary when the picture height in luma samples is not an integer multiple of the luma CTB size

– The division of each coding unit into prediction units

– The division of each coding unit into transform units

– The division of each coding unit into coding blocks

– The division of each coding block into prediction blocks

– The division of each coding block into transform blocks

– The division of each prediction unit into prediction blocks

– The division of each transform unit into transform blocks.

6.4 Availability processes

6.4.1 Derivation process for z-scan order block availability

Inputs to this process are:

– The luma location (xCurr, yCurr) of the top-left sample of the current block relative to the top-left luma sample of the current picture

– The luma location (xNbY, yNbY) covered by a neighbouring block relative to the top-left luma sample of the current picture.

Output of this process is the availability of the neighbouring block covering the location (xNbY, yNbY), denoted as availableN.

The minimum luma block address in z-scan order minBlockAddrCurr of the current block is derived as follows:

minBlockAddrCurr = MinTbAddrZs[xCurr >> MinTbLog2SizeY][yCurr >> MinTbLog2SizeY] (6-1)

The minimum luma block address in z-scan order minBlockAddrN of the neighbouring block covering the location (xNbY, yNbY) is derived as follows:

– If one or more of the following conditions are true, minBlockAddrN is set equal to −1:
  – xNbY is less than 0
  – yNbY is less than 0
  – xNbY is greater than or equal to pic_width_in_luma_samples
  – yNbY is greater than or equal to pic_height_in_luma_samples
– Otherwise (xNbY and yNbY are inside the picture boundaries),

minBlockAddrN = MinTbAddrZs[xNbY >> MinTbLog2SizeY][yNbY >> MinTbLog2SizeY] (6-2)

The neighbouring block availability availableN is derived as follows:

– If one or more of the following conditions are true, availableN is set equal to FALSE:
  – minBlockAddrN is less than 0,
  – minBlockAddrN is greater than minBlockAddrCurr,
  – the variable SliceAddrRs associated with the slice segment containing the neighbouring block with the minimum luma block address minBlockAddrN differs in value from the variable SliceAddrRs associated with the slice segment containing the current block with the minimum luma block address minBlockAddrCurr.
  – the neighbouring block with the minimum luma block address minBlockAddrN is contained in a different tile than the current block with the minimum luma block address minBlockAddrCurr.
– Otherwise, availableN is set equal to TRUE.

6.4.2 Derivation process for prediction block availability

Inputs to this process are:

– the luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– the luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the current luma prediction block,

– a variable partIdx specifying the partition index of the current prediction unit within the current coding unit,

– the luma location (xNbY, yNbY) covered by a neighbouring prediction block relative to the top-left luma sample of the current picture.

Output of this process is the availability of the neighbouring prediction block covering the location (xNbY, yNbY), denoted as availableN is derived as follows:

The variable sameCb identifies whether the current luma prediction block and the neighbouring luma prediction block cover the same luma coding block, and is derived as follows:

– If all of the following conditions are true, sameCb is set equal to TRUE:
  – xCb is less than or equal than xNbY,
  – yCb is less than or equal than yNbY,
  – (xCb + nCbS) is greater than xNbY,
  – (yCb + nCbS) is greater than yNbY.
– Otherwise, sameCb is set equal to FALSE.

The neighbouring prediction block availability availableN is derived as follows:

– If sameCb is equal to FALSE, the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to (xPb, yPb) and the luma location (xNbY, yNbY) as inputs, and the output is assigned to availableN.
– Otherwise, if all of the following conditions are true, availableN is set equal to FALSE:
  – (nPbW << 1) is equal to nCbS,
  – (nPbH << 1) is equal to nCbS,
  – partIdx is equal to 1,
  – (yCb + nPbH) is less than or equal to yNbY,
  – (xCb + nPbW) is greater than xNbY.
– Otherwise, availableN is set equal to TRUE.

When availableN is equal to TRUE and CuPredMode[xNbY][yNbY] is equal to MODE_INTRA, availableN is set equal to FALSE.

6.5 Scanning processes

6.5.1 CTB raster and tile scanning conversion process

The list colWidth[i] for i ranging from 0 to num_tile_columns_minus1, inclusive, specifying the width of the i-th tile column in units of CTBs, is derived as follows:

```
if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 ) −
                        ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
else {
    colWidth[ num_tile_columns_minus1 ] = PicWidthInCtbsY                          (6-3)
    for( i = 0; i < num_tile_columns_minus1; i++ ) {
        colWidth[ i ] = column_width_minus1[ i ] + 1
        colWidth[ num_tile_columns_minus1 ] −= colWidth[ i ]
    }
}
```

The list rowHeight[j] for j ranging from 0 to num_tile_rows_minus1, inclusive, specifying the height of the j-th tile row in units of CTBs, is derived as follows:

```
if( uniform_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        rowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 ) −
                         ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
else {
    rowHeight[ num_tile_rows_minus1 ] = PicHeightInCtbsY                           (6-4)
```

```
    for( j = 0; j < num_tile_rows_minus1; j++ ) {
        rowHeight[ j ] = row_height_minus1[ j ] + 1
        rowHeight[ num_tile_rows_minus1 ] −= rowHeight[ j ]
    }
}
```

The list colBd[i] for i ranging from 0 to num_tile_columns_minus1 + 1, inclusive, specifying the location of the i-th tile column boundary in units of CTBs, is derived as follows:

```
for( colBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
        colBd[ i + 1 ] = colBd[ i ] + colWidth[ i ]
    (6-5)
```

The list rowBd[j] for j ranging from 0 to num_tile_rows_minus1 + 1, inclusive, specifying the location of the j-th tile row boundary in units of CTBs, is derived as follows:

```
for( rowBd[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )
    rowBd[ j + 1 ] = rowBd[ j ] + rowHeight[ j ]
    (6-6)
```

The list CtbAddrRsToTs[ctbAddrRs] for ctbAddrRs ranging from 0 to PicSizeInCtbsY − 1, inclusive, specifying the conversion from a CTB address in CTB raster scan of a picture to a CTB address in tile scan, is derived as follows:

```
for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) {
    tbX = ctbAddrRs % PicWidthInCtbsY
    tbY = ctbAddrRs / PicWidthInCtbsY
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        if( tbX >= colBd[ i ] )
            tileX = i
    for( j = 0; j <= num_tile_rows_minus1; j++ )
    (6-7)
        if( tbY >= rowBd[ j ] )
            tileY = j
    CtbAddrRsToTs[ ctbAddrRs ] = 0
    for( i = 0; i < tileX; i++ )
        CtbAddrRsToTs[ ctbAddrRs ] += rowHeight[ tileY ] * colWidth[ i ]
    for( j = 0; j < tileY; j++ )
        CtbAddrRsToTs[ ctbAddrRs ] += PicWidthInCtbsY * rowHeight[ j ]
    CtbAddrRsToTs[ ctbAddrRs ] += ( tbY − rowBd[ tileY ]
) * colWidth[ tileX ] + tbX − colBd[ tileX ]
}
```

The list CtbAddrTsToRs[ctbAddrTs] for ctbAddrTs ranging from 0 to PicSizeInCtbsY − 1, inclusive, specifying the conversion from a CTB address in tile scan to a CTB address in CTB raster scan of a picture, is derived as follows:

```
for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ )                    (6-8)
    CtbAddrTsToRs[ CtbAddrRsToTs[ ctbAddrRs ] ] = ctbAddrRs
```

The list TileId[ctbAddrTs] for ctbAddrTs ranging from 0 to PicSizeInCtbsY − 1, inclusive, specifying the conversion from a CTB address in tile scan to a tile ID, is derived as follows:

```
for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )
        for( y = rowBd[ j ]; y < rowBd[ j + 1 ]; y++ )                              (6-9)
            for( x = colBd[ i ]; x < colBd[ i + 1 ]; x++ )
                    TileId[ CtbAddrRsToTs[ y * PicWidthInCtbsY+ x ] ] = tileIdx
```

The values of ColumnWidthInLumaSamples[i], specifying the width of the i-th tile column in units of luma samples, are set equal to colWidth[i] << CtbLog2SizeY for i ranging from 0 to num_tile_columns_minus1, inclusive.

The values of RowHeightInLumaSamples[j], specifying the height of the j-th tile row in units of luma samples, are set equal to rowHeight[j] << CtbLog2SizeY for j ranging from 0 to num_tile_rows_minus1, inclusive.

6.5.2 Z-scan order array initialization process

The array MinTbAddrZs with elements MinTbAddrZs[x][y] for x ranging from 0 to (PicWidthInCtbsY << (CtbLog2SizeY − MinTbLog2SizeY)) − 1, inclusive, and y ranging from 0 to (PicHeightInCtbsY << (CtbLog2SizeY − MinTbLog2SizeY)) − 1, specifying the conversion from a location (x, y) in units of minimum blocks to a minimum block address in z-scan order, inclusive, is derived as follows:

```
for( y = 0; y < ( PicHeightInCtbsY << ( CtbLog2SizeY − MinTbLog2SizeY ) ); y++ )
   for( x = 0; x < ( PicWidthInCtbsY << ( CtbLog2SizeY − MinTbLog2SizeY ) ); x++) {
      tbX = ( x << MinTbLog2SizeY ) >> CtbLog2SizeY
      tbY = ( y << MinTbLog2SizeY ) >> CtbLog2SizeY
      ctbAddrRs = PicWidthInCtbsY * tbY + tbX
      MinTbAddrZs[ x ][ y ] = CtbAddrRsToTs[ ctbAddrRs ] <<                    (6-10)
            ( ( CtbLog2SizeY − MinTbLog2SizeY ) * 2 )
      for( i = 0, p = 0; i < ( CtbLog2SizeY − MinTbLog2SizeY ); i++ ) {
         m = 1 << i
         p += ( m & x ? m * m : 0 ) + ( m & y ? 2 * m * m : 0 )
      }
      MinTbAddrZs[ x ][ y ] += p
   }
```

6.5.3 Up-right diagonal scan order array initialization process

Input to this process is a block size blkSize.

Output of this process is the array diagScan[sPos][sComp]. The array index sPos specify the scan position ranging from 0 to (blkSize * blkSize) − 1. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. Depending on the value of blkSize, the array diagScan is derived as follows:

```
i = 0
x = 0
y = 0
stopLoop = FALSE
while( !stopLoop ) {
   while( y >= 0 ) {
      if( x < blkSize && y < blkSize ) {                                   (6-11)
         diagScan[ i ][ 0 ] = x
         diagScan[ i ][ 1 ] = y
         i++
      }
      y− −
      x++
   }
   y = x
   x = 0
   if( i >= blkSize * blkSize )
      stopLoop = TRUE
}
```

6.5.4 Horizontal scan order array initialization process

Input to this process is a block size blkSize.

Output of this process is the array horScan[sPos][sComp]. The array index sPos specifies the scan position ranging from 0 to (blkSize * blkSize) − 1. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. Depending on the value of blkSize, the array horScan is derived as follows:

```
i = 0
for( y = 0; y < blkSize; y++ )
    for( x = 0; x < blkSize; x++ ) {
        horScan[ i ][ 0 ] = x                                        (6-12)
        horScan[ i ][ 1 ] = y
        i++
    }
```

6.5.5 Vertical scan order array initialization process

Input to this process is a block size blkSize.

Output of this process is the array verScan[sPos][sComp]. The array index sPos specifies the scan position ranging from 0 to (blkSize * blkSize) − 1. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. Depending on the value of blkSize, the array verScan is derived as follows:

```
i = 0
for( x = 0; x < blkSize; x++ )
    for( y = 0; y < blkSize; y++ ) {
        verScan[ i ][ 0 ] = x                                        (6-13)
        verScan[ i ][ 1 ] = y
        i++
    }
```

6.5.6 Traverse scan order array initialization process

Input to this process is a block size blkSize.

Output of this process is the array travScan[sPos][sComp]. The array index sPos specifies the scan position ranging from 0 to (blkSize * blkSize) − 1, inclusive. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. Depending on the value of blkSize, the array travScan is derived as follows:

```
i = 0
for( y = 0; y < blkSize; y++ )
    if( y % 2 = = 0 )
        for( x = 0; x < blkSize; x++ ) {
            travScan[ i ][ 0 ] = x
            travScan[ i ][ 1 ] = y
            i++
        }
    else                                                             (6-14)
        for( x = blkSize − 1; x >= 0; x− − ) {
            travScan[ i ][ 0 ] = x
            travScan[ i ][ 1 ] = y
            i++
        }
```

7 Syntax and semantics

7.1 Method of specifying syntax in tabular form

The syntax tables specify a superset of the syntax of all allowed bitstreams. Additional constraints on the syntax may be specified, either directly or indirectly, in other clauses.

NOTE – An actual decoder should implement some means for identifying entry points into the bitstream and some means to identify and handle non-conforming bitstreams. The methods for identifying and handling errors and other such situations are not specified in this Specification.

The following table lists examples of the syntax specification format. When **syntax_element** appears, it specifies that a syntax element is parsed from the bitstream and the bitstream pointer is advanced to the next position beyond the syntax element in the bitstream parsing process.

	Descriptor
/* A statement can be a syntax element with an associated descriptor or can be an expression used to specify conditions for the existence, type and quantity of syntax elements, as in the following two examples */	
syntax_element	ue(v)
conditioning statement	
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */	
{	
statement	
statement	
...	
}	
/* A "while" structure specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */	
while(condition)	
statement	
/* A "do ... while" structure specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true */	
do	
statement	
while(condition)	
/* An "if ... else" structure specifies a test of whether a condition is true and, if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */	
if(condition)	
primary statement	
else	
alternative statement	
/* A "for" structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */	
for(initial statement; condition; subsequent statement)	
primary statement	

7.2 Specification of syntax functions and descriptors

The functions presented here are used in the syntactical description. These functions are expressed in terms of the value of a bitstream pointer that indicates the position of the next bit to be read by the decoding process from the bitstream.

byte_aligned( ) is specified as follows:

– If the current position in the bitstream is on a byte boundary, i.e., the next bit in the bitstream is the first bit in a byte, the return value of byte_aligned( ) is equal to TRUE.

– Otherwise, the return value of byte_aligned( ) is equal to FALSE.

more_data_in_byte_stream( ), which is used only in the byte stream NAL unit syntax structure specified in Annex B, is specified as follows:

– If more data follow in the byte stream, the return value of more_data_in_byte_stream( ) is equal to TRUE.

– Otherwise, the return value of more_data_in_byte_stream( ) is equal to FALSE.

more_data_in_payload( ) is specified as follows:

– If byte_aligned( ) is equal to TRUE and the current position in the sei_payload( ) syntax structure is 8 * payloadSize bits from the beginning of the sei_payload( ) syntax structure, the return value of more_data_in_payload( ) is equal to FALSE.

– Otherwise, the return value of more_data_in_payload( ) is equal to TRUE.

more_rbsp_data( ) is specified as follows:

– If there is no more data in the raw byte sequence payload (RBSP), the return value of more_rbsp_data( ) is equal to FALSE.

– Otherwise, the RBSP data are searched for the last (least significant, right-most) bit equal to 1 that is present in the RBSP. Given the position of this bit, which is the first bit (rbsp_stop_one_bit) of the rbsp_trailing_bits( ) syntax structure, the following applies:

  – If there is more data in an RBSP before the rbsp_trailing_bits( ) syntax structure, the return value of more_rbsp_data( ) is equal to TRUE.

  – Otherwise, the return value of more_rbsp_data( ) is equal to FALSE.

  The method for enabling determination of whether there is more data in the RBSP is specified by the application (or in Annex B for applications that use the byte stream format).

more_rbsp_trailing_data( ) is specified as follows:

– If there is more data in an RBSP, the return value of more_rbsp_trailing_data( ) is equal to TRUE.

– Otherwise, the return value of more_rbsp_trailing_data( ) is equal to FALSE.

next_bits(n) provides the next bits in the bitstream for comparison purposes, without advancing the bitstream pointer. It provides a look at the next n bits in the bitstream with n being its argument. When used within the byte stream format as specified in Annex B and fewer than n bits remain within the byte stream, next_bits(n) returns a value of 0.

payload_extension_present( ) is specified as follows:

– If the current position in the sei_payload( ) syntax structure is not the position of the last (least significant, right-most) bit that is equal to 1 that is less than 8 * payloadSize bits from the beginning of the syntax structure (i.e., the position of the payload_bit_equal_to_one syntax element), the return value of payload_extension_present( ) is equal to TRUE.

– Otherwise, the return value of payload_extension_present( ) is equal to FALSE.

pic_layer_id(picX) returns the value of the nuh_layer_id of the VCL NAL units in the picture picX.

read_bits(n) reads the next n bits from the bitstream and advances the bitstream pointer by n bit positions. When n is equal to 0, read_bits(n) is specified to return a value equal to 0 and to not advance the bitstream pointer.

The following descriptors specify the parsing process of each syntax element:

– ae(v): context-adaptive arithmetic entropy-coded syntax element. The parsing process for this descriptor is specified in clause 9.3.

– b(8): byte having any pattern of bit string (8 bits). The parsing process for this descriptor is specified by the return value of the function read_bits(8).

– f(n): fixed-pattern bit string using n bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function read_bits(n).

– i(n): signed integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a two's complement integer representation with most significant bit written first.

– se(v): signed integer 0-th order Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in clause 9.2.

– st(v): null-terminated string encoded as universal coded character set (UCS) transmission format-8 (UTF-8) characters as specified in ISO/IEC 10646. The parsing process is specified as follows: st(v) begins at a byte-aligned position in the bitstream and reads and returns a series of bytes from the bitstream, beginning at the current position and continuing up to but not including the next byte-aligned byte that is equal to 0x00, and advances the bitstream pointer by (stringLength + 1) * 8 bit positions, where stringLength is equal to the number of bytes returned.

  NOTE – The st(v) syntax descriptor is only used in this Specification when the current position in the bitstream is a byte-aligned position.

– u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a binary representation of an unsigned integer with most significant bit written first.

– ue(v): unsigned integer 0-th order Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in clause 9.2.

7.3 Syntax in tabular form

7.3.1 NAL unit syntax

7.3.1.1 General NAL unit syntax

nal_unit(NumBytesInNalUnit) {	**Descriptor**
nal_unit_header( )	
NumBytesInRbsp = 0	
for(i = 2; i < NumBytesInNalUnit; i++)	
if(i + 2 < NumBytesInNalUnit && next_bits(24) = = 0x000003) {	
rbsp_byte[NumBytesInRbsp++]	b(8)
rbsp_byte[NumBytesInRbsp++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRbsp++]	b(8)
}	

7.3.1.2 NAL unit header syntax

nal_unit_header( ) {	**Descriptor**
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nuh_layer_id	u(6)
nuh_temporal_id_plus1	u(3)
}	

7.3.2 Raw byte sequence payloads, trailing bits and byte alignment syntax

7.3.2.1 Video parameter set RBSP syntax

video_parameter_set_rbsp( ) {	**Descriptor**
vps_video_parameter_set_id	u(4)
vps_base_layer_internal_flag	u(1)
vps_base_layer_available_flag	u(1)
vps_max_layers_minus1	u(6)
vps_max_sub_layers_minus1	u(3)
vps_temporal_id_nesting_flag	u(1)
vps_reserved_0xffff_16bits	u(16)
profile_tier_level(1, vps_max_sub_layers_minus1)	
vps_sub_layer_ordering_info_present_flag	u(1)
for(i = (vps_sub_layer_ordering_info_present_flag ? 0 : vps_max_sub_layers_minus1); i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering_minus1[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase_plus1[i]	ue(v)
}	
vps_max_layer_id	u(6)
vps_num_layer_sets_minus1	ue(v)
for(i = 1; i <= vps_num_layer_sets_minus1; i++)	
for(j = 0; j <= vps_max_layer_id; j++)	
layer_id_included_flag[i][j]	u(1)
vps_timing_info_present_flag	u(1)
if(vps_timing_info_present_flag) {	
vps_num_units_in_tick	u(32)
vps_time_scale	u(32)
vps_poc_proportional_to_timing_flag	u(1)
if(vps_poc_proportional_to_timing_flag)	
vps_num_ticks_poc_diff_one_minus1	ue(v)
vps_num_hrd_parameters	ue(v)
for(i = 0; i < vps_num_hrd_parameters; i++) {	
hrd_layer_set_idx[i]	ue(v)
if(i > 0)	
cprms_present_flag[i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
}	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

7.3.2.2 Sequence parameter set RBSP syntax

7.3.2.2.1 General sequence parameter set RBSP syntax

seq_parameter_set_rbsp( ) {	**Descriptor**
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc = = 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)
if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
log2_min_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_luma_coding_block_size	ue(v)
log2_min_luma_transform_block_size_minus2	ue(v)
log2_diff_max_min_luma_transform_block_size	ue(v)
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enabled_flag	u(1)
if(scaling_list_enabled_flag) {	
sps_scaling_list_data_present_flag	u(1)
if(sps_scaling_list_data_present_flag)	
scaling_list_data( )	
}	
amp_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_sample_bit_depth_luma_minus1	u(4)
pcm_sample_bit_depth_chroma_minus1	u(4)

log2_min_pcm_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_luma_coding_block_size	ue(v)
pcm_loop_filter_disabled_flag	u(1)
}	
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
st_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
if(long_term_ref_pics_present_flag) {	
num_long_term_ref_pics_sps	ue(v)
for(i = 0; i < num_long_term_ref_pics_sps; i++) {	
lt_ref_pic_poc_lsb_sps[i]	u(v)
used_by_curr_pic_lt_sps_flag[i]	u(1)
}	
}	
sps_temporal_mvp_enabled_flag	u(1)
strong_intra_smoothing_enabled_flag	u(1)
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
sps_extension_present_flag	u(1)
if(sps_extension_present_flag) {	
sps_range_extension_flag	u(1)
sps_multilayer_extension_flag	u(1)
sps_3d_extension_flag	u(1)
sps_scc_extension_flag	u(1)
sps_extension_4bits	u(4)
}	
if(sps_range_extension_flag)	
sps_range_extension( )	
if(sps_multilayer_extension_flag)	
sps_multilayer_extension( ) /* specified in Annex F */	
if(sps_3d_extension_flag)	
sps_3d_extension( ) /* specified in Annex I */	
if(sps_scc_extension_flag)	
sps_scc_extension( )	
if(sps_extension_4bits)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

7.3.2.2.2 Sequence parameter set range extension syntax

sps_range_extension( ) {	**Descriptor**
transform_skip_rotation_enabled_flag	u(1)
transform_skip_context_enabled_flag	u(1)
implicit_rdpcm_enabled_flag	u(1)
explicit_rdpcm_enabled_flag	u(1)
extended_precision_processing_flag	u(1)
intra_smoothing_disabled_flag	u(1)
high_precision_offsets_enabled_flag	u(1)
persistent_rice_adaptation_enabled_flag	u(1)
cabac_bypass_alignment_enabled_flag	u(1)
}	

7.3.2.2.3 Sequence parameter set screen content coding extension syntax

sps_scc_extension( ) {	**Descriptor**
sps_curr_pic_ref_enabled_flag	u(1)
palette_mode_enabled_flag	u(1)
if(palette_mode_enabled_flag) {	
palette_max_size	ue(v)
delta_palette_max_predictor_size	ue(v)
sps_palette_predictor_initializers_present_flag	u(1)
if(sps_palette_predictor_initializers_present_flag) {	
sps_num_palette_predictor_initializers_minus1	ue(v)
numComps = (chroma_format_idc = = 0) ? 1 : 3	
for(comp = 0; comp < numComps; comp++)	
for(i = 0; i <= sps_num_palette_predictor_initializers_minus1; i++)	
sps_palette_predictor_initializer[comp][i]	u(v)
}	
}	
motion_vector_resolution_control_idc	u(2)
intra_boundary_filtering_disabled_flag	u(1)
}	

7.3.2.3 Picture parameter set RBSP syntax

7.3.2.3.1 General picture parameter set RBSP syntax

pic_parameter_set_rbsp( ) {	**Descriptor**
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
dependent_slice_segments_enabled_flag	u(1)
output_flag_present_flag	u(1)
num_extra_slice_header_bits	u(3)
sign_data_hiding_enabled_flag	u(1)
cabac_init_present_flag	u(1)

num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
transform_skip_enabled_flag	u(1)
cu_qp_delta_enabled_flag	u(1)
if(cu_qp_delta_enabled_flag)	
diff_cu_qp_delta_depth	ue(v)
pps_cb_qp_offset	se(v)
pps_cr_qp_offset	se(v)
pps_slice_chroma_qp_offsets_present_flag	u(1)
weighted_pred_flag	u(1)
weighted_bipred_flag	u(1)
transquant_bypass_enabled_flag	u(1)
tiles_enabled_flag	u(1)
entropy_coding_sync_enabled_flag	u(1)
if(tiles_enabled_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
pps_loop_filter_across_slices_enabled_flag	u(1)
deblocking_filter_control_present_flag	u(1)
if(deblocking_filter_control_present_flag) {	
deblocking_filter_override_enabled_flag	u(1)
pps_deblocking_filter_disabled_flag	u(1)
if(!pps_deblocking_filter_disabled_flag) {	
pps_beta_offset_div2	se(v)
pps_tc_offset_div2	se(v)
}	
}	
pps_scaling_list_data_present_flag	u(1)
if(pps_scaling_list_data_present_flag)	
scaling_list_data( )	
lists_modification_present_flag	u(1)
log2_parallel_merge_level_minus2	ue(v)
slice_segment_header_extension_present_flag	u(1)
pps_extension_present_flag	u(1)
if(pps_extension_present_flag) {	
pps_range_extension_flag	u(1)
pps_multilayer_extension_flag	u(1)

pps_3d_extension_flag	u(1)
pps_scc_extension_flag	u(1)
pps_extension_4bits	u(4)
}	
if(pps_range_extension_flag)	
pps_range_extension( )	
if(pps_multilayer_extension_flag)	
pps_multilayer_extension( ) /* specified in Annex F */	
if(pps_3d_extension_flag)	
pps_3d_extension( ) /* specified in Annex I */	
if(pps_scc_extension_flag)	
pps_scc_extension( )	
if(pps_extension_4bits)	
while(more_rbsp_data( ))	
pps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

7.3.2.3.2 Picture parameter set range extension syntax

pps_range_extension( ) {	**Descriptor**
if(transform_skip_enabled_flag)	
log2_max_transform_skip_block_size_minus2	ue(v)
cross_component_prediction_enabled_flag	u(1)
chroma_qp_offset_list_enabled_flag	u(1)
if(chroma_qp_offset_list_enabled_flag) {	
diff_cu_chroma_qp_offset_depth	ue(v)
chroma_qp_offset_list_len_minus1	ue(v)
for(i = 0; i <= chroma_qp_offset_list_len_minus1; i++) {	
cb_qp_offset_list[i]	se(v)
cr_qp_offset_list[i]	se(v)
}	
}	
log2_sao_offset_scale_luma	ue(v)
log2_sao_offset_scale_chroma	ue(v)
}	

7.3.2.3.3 Picture parameter set screen content coding extension syntax

pps_scc_extension( ) {	**Descriptor**
pps_curr_pic_ref_enabled_flag	u(1)
residual_adaptive_colour_transform_enabled_flag	u(1)
if(residual_adaptive_colour_transform_enabled_flag) {	
pps_slice_act_qp_offsets_present_flag	u(1)
pps_act_y_qp_offset_plus5	se(v)
pps_act_cb_qp_offset_plus5	se(v)
pps_act_cr_qp_offset_plus3	se(v)
}	
pps_palette_predictor_initializers_present_flag	u(1)
if(pps_palette_predictor_initializers_present_flag) {	
pps_num_palette_predictor_initializers	ue(v)
if(pps_num_palette_predictor_initializers > 0) {	
monochrome_palette_flag	u(1)
luma_bit_depth_entry_minus8	ue(v)
if(!monochrome_palette_flag)	
chroma_bit_depth_entry_minus8	ue(v)
numComps = monochrome_palette_flag ? 1 : 3	
for(comp = 0; comp < numComps; comp++)	
for(i = 0; i < pps_num_palette_predictor_initializers; i++)	
pps_palette_predictor_initializer[comp][i]	u(v)
}	
}	
}	

7.3.2.4 Supplemental enhancement information RBSP syntax

sei_rbsp( ) {	**Descriptor**
do	
sei_message( )	
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

7.3.2.5 Access unit delimiter RBSP syntax

access_unit_delimiter_rbsp( ) {	**Descriptor**
pic_type	u(3)
rbsp_trailing_bits( )	
}	

7.3.2.6 End of sequence RBSP syntax

end_of_seq_rbsp( ) {	**Descriptor**
}	

7.3.2.7 End of bitstream RBSP syntax

end_of_bitstream_rbsp( ) {	**Descriptor**
}	

7.3.2.8 Filler data RBSP syntax

filler_data_rbsp( ) {	**Descriptor**
while(next_bits(8) = = 0xFF)	
ff_byte /* equal to 0xFF */	f(8)
rbsp_trailing_bits( )	
}	

7.3.2.9 Slice segment layer RBSP syntax

slice_segment_layer_rbsp( ) {	**Descriptor**
slice_segment_header( )	
slice_segment_data( )	
rbsp_slice_segment_trailing_bits( )	
}	

7.3.2.10 RBSP slice segment trailing bits syntax

rbsp_slice_segment_trailing_bits( ) {	**Descriptor**
rbsp_trailing_bits( )	
while(more_rbsp_trailing_data( ))	
cabac_zero_word /* equal to 0x0000 */	f(16)
}	

7.3.2.11 RBSP trailing bits syntax

rbsp_trailing_bits( ) {	**Descriptor**
rbsp_stop_one_bit /* equal to 1 */	f(1)
while(!byte_aligned( ))	
rbsp_alignment_zero_bit /* equal to 0 */	f(1)
}	

7.3.2.12 Byte alignment syntax

byte_alignment( ) {	**Descriptor**
alignment_bit_equal_to_one /* equal to 1 */	f(1)
while(!byte_aligned( ))	
alignment_bit_equal_to_zero /* equal to 0 */	f(1)
}	

7.3.3 Profile, tier and level syntax

profile_tier_level(profilePresentFlag, maxNumSubLayersMinus1) {	**Descriptor**
if(profilePresentFlag) {	
general_profile_space	u(2)
general_tier_flag	u(1)
general_profile_idc	u(5)
for(j = 0; j < 32; j++)	
general_profile_compatibility_flag[j]	u(1)
general_progressive_source_flag	u(1)
general_interlaced_source_flag	u(1)
general_non_packed_constraint_flag	u(1)
general_frame_only_constraint_flag	u(1)

if(general_profile_idc = = 4 || general_profile_compatibility_flag[4] ||
general_profile_idc = = 5 || general_profile_compatibility_flag[5] ||
general_profile_idc = = 6 || general_profile_compatibility_flag[6] ||
general_profile_idc = = 7 || general_profile_compatibility_flag[7] ||
general_profile_idc = = 8 || general_profile_compatibility_flag[8] ||
general_profile_idc = = 9 || general_profile_compatibility_flag[9] ||
general_profile_idc = = 10 || general_profile_compatibility_flag[10] ||
general_profile_idc = = 11 || general_profile_compatibility_flag[11]) {
/* The number of bits in this syntax structure is not affected by this condition */	

general_max_12bit_constraint_flag	u(1)
general_max_10bit_constraint_flag	u(1)
general_max_8bit_constraint_flag	u(1)
general_max_422chroma_constraint_flag	u(1)
general_max_420chroma_constraint_flag	u(1)
general_max_monochrome_constraint_flag	u(1)
general_intra_constraint_flag	u(1)
general_one_picture_only_constraint_flag	u(1)
general_lower_bit_rate_constraint_flag	u(1)

if(general_profile_idc = = 5 || general_profile_compatibility_flag[5] ||
general_profile_idc = = 9 || general_profile_compatibility_flag[9] ||
general_profile_idc = = 10 || general_profile_compatibility_flag[10] ||
general_profile_idc = = 11 || general_profile_compatibility_flag[11]) {	

general_max_14bit_constraint_flag	u(1)
general_reserved_zero_33bits	u(33)
} else	
general_reserved_zero_34bits	u(34)
} else if(general_profile_idc = = 2 || general_profile_compatibility_flag[2]) {	
general_reserved_zero_7bits	u(7)
general_one_picture_only_constraint_flag	u(1)

general_reserved_zero_35bits	u(35)
} else	
general_reserved_zero_43bits	u(43)
if(general_profile_idc = = 1 || general_profile_compatibility_flag[1] || general_profile_idc = = 2 || general_profile_compatibility_flag[2] || general_profile_idc = = 3 || general_profile_compatibility_flag[3] || general_profile_idc = = 4 || general_profile_compatibility_flag[4] || general_profile_idc = = 5 || general_profile_compatibility_flag[5] || general_profile_idc = = 9 || general_profile_compatibility_flag[9] || general_profile_idc = = 11 || general_profile_compatibility_flag[11]) /* The number of bits in this syntax structure is not affected by this condition */	
general_inbld_flag	u(1)
else	
general_reserved_zero_bit	u(1)
}	
general_level_idc	u(8)
for(i = 0; i < maxNumSubLayersMinus1; i++) {	
sub_layer_profile_present_flag[i]	u(1)
sub_layer_level_present_flag[i]	u(1)
}	
if(maxNumSubLayersMinus1 > 0)	
for(i = maxNumSubLayersMinus1; i < 8; i++)	
reserved_zero_2bits[i]	u(2)
for(i = 0; i < maxNumSubLayersMinus1; i++) {	
if(sub_layer_profile_present_flag[i]) {	
sub_layer_profile_space[i]	u(2)
sub_layer_tier_flag[i]	u(1)
sub_layer_profile_idc[i]	u(5)
for(j = 0; j < 32; j++)	
sub_layer_profile_compatibility_flag[i][j]	u(1)
sub_layer_progressive_source_flag[i]	u(1)
sub_layer_interlaced_source_flag[i]	u(1)
sub_layer_non_packed_constraint_flag[i]	u(1)
sub_layer_frame_only_constraint_flag[i]	u(1)

if(sub_layer_profile_idc[i] = = 4 ||
sub_layer_profile_compatibility_flag[i][4] ||
sub_layer_profile_idc[i] = = 5 ||
sub_layer_profile_compatibility_flag[i][5] ||
sub_layer_profile_idc[i] = = 6 ||
sub_layer_profile_compatibility_flag[i][6] ||
sub_layer_profile_idc[i] = = 7 ||
sub_layer_profile_compatibility_flag[i][7] ||
sub_layer_profile_idc[i] = = 8 ||
sub_layer_profile_compatibility_flag[i][8] ||
sub_layer_profile_idc[i] = = 9 ||
sub_layer_profile_compatibility_flag[i][9] ||
sub_layer_profile_idc[i] = = 10 ||
sub_layer_profile_compatibility_flag[i][10] ||
sub_layer_profile_idc[i] = = 11 ||
sub_layer_profile_compatibility_flag[i][11]) {
/* The number of bits in this syntax structure is not affected by this condition */	

sub_layer_max_12bit_constraint_flag[i]	u(1)
sub_layer_max_10bit_constraint_flag[i]	u(1)
sub_layer_max_8bit_constraint_flag[i]	u(1)

sub_layer_max_422chroma_constraint_flag[i]	u(1)
sub_layer_max_420chroma_constraint_flag[i]	u(1)
sub_layer_max_monochrome_constraint_flag[i]	u(1)
sub_layer_intra_constraint_flag[i]	u(1)
sub_layer_one_picture_only_constraint_flag[i]	u(1)
sub_layer_lower_bit_rate_constraint_flag[i]	u(1)

if(sub_layer_profile_idc[i] = = 5 ||
sub_layer_profile_compatibility_flag[i][5] ||
sub_layer_profile_idc[i] = = 9 ||
sub_layer_profile_compatibility_flag[i][9] ||
sub_layer_profile_idc[i] = = 10 ||
sub_layer_profile_compatibility_flag[i][10] ||
sub_layer_profile_idc[i] = = 11 ||
sub_layer_profile_compatibility_flag[i][11]) {	

sub_layer_max_14bit_constraint_flag[i]	u(1)
sub_layer_reserved_zero_33bits[i]	u(33)
} else	
sub_layer_reserved_zero_34bits[i]	u(34)
} else if(sub_layer_profile_idc[i] = = 2 || sub_layer_profile_compatibility_flag[i][2]) {	
sub_layer_reserved_zero_7bits[i]	u(7)
sub_layer_one_picture_only_constraint_flag[i]	u(1)
sub_layer_reserved_zero_35bits[i]	u(35)
} else	
sub_layer_reserved_zero_43bits[i]	u(43)
if(sub_layer_profile_idc[i] = = 1 || sub_layer_profile_compatibility_flag[i][1] || sub_layer_profile_idc[i] = = 2 || sub_layer_profile_compatibility_flag[i][2] || sub_layer_profile_idc[i] = = 3 || sub_layer_profile_compatibility_flag[i][3] || sub_layer_profile_idc[i] = = 4 || sub_layer_profile_compatibility_flag[i][4] || sub_layer_profile_idc[i] = = 5 || sub_layer_profile_compatibility_flag[i][5] || sub_layer_profile_idc[i] = = 9 || sub_layer_profile_compatibility_flag[i][9] || sub_layer_profile_idc[i] = = 11 || sub_layer_profile_compatibility_flag[i][11]) /* The number of bits in this syntax structure is not affected by this condition */	
sub_layer_inbld_flag[i]	u(1)
else	
sub_layer_reserved_zero_bit[i]	u(1)
}	
if(sub_layer_level_present_flag[i])	
sub_layer_level_idc[i]	u(8)
}	
}	

7.3.4 Scaling list data syntax

scaling_list_data( ) {	**Descriptor**
for(sizeId = 0; sizeId < 4; sizeId++)	
for(matrixId = 0; matrixId < 6; matrixId += (sizeId = = 3) ? 3 : 1) {	
scaling_list_pred_mode_flag[sizeId][matrixId]	u(1)
if(!scaling_list_pred_mode_flag[sizeId][matrixId])	
scaling_list_pred_matrix_id_delta[sizeId][matrixId]	ue(v)
else {	
nextCoef = 8	
coefNum = Min(64, (1 << (4 + (sizeId << 1))))	
if(sizeId > 1) {	
scaling_list_dc_coef_minus8[sizeId − 2][matrixId]	se(v)
nextCoef = scaling_list_dc_coef_minus8[sizeId − 2][matrixId] + 8	
}	
for(i = 0; i < coefNum; i++) {	
scaling_list_delta_coef	se(v)
nextCoef = (nextCoef + scaling_list_delta_coef + 256) % 256	
ScalingList[sizeId][matrixId][i] = nextCoef	
}	
}	
}	
}	

7.3.5 Supplemental enhancement information message syntax

sei_message( ) {	**Descriptor**
payloadType = 0	
while(next_bits(8) = = 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadType += 255	
}	
last_payload_type_byte	u(8)
payloadType += last_payload_type_byte	
payloadSize = 0	
while(next_bits(8) = = 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadSize += 255	
}	
last_payload_size_byte	u(8)
payloadSize += last_payload_size_byte	
sei_payload(payloadType, payloadSize)	
}	

7.3.6 Slice segment header syntax

7.3.6.1 General slice segment header syntax

slice_segment_header( ) {	**Descriptor**
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
CuQpDeltaVal = 0	
if(!dependent_slice_segment_flag) {	
for(i = 0; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
slice_pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
st_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)

}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(slice_type = = P || slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification( )	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_flag && slice_type = = B))	
pred_weight_table( )	
five_minus_max_num_merge_cand	ue(v)
if(motion_vector_resolution_control_idc = = 2)	
use_integer_mv_flag	u(1)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(pps_slice_act_qp_offsets_present_flag) {	
slice_act_y_qp_offset	se(v)
slice_act_cb_qp_offset	se(v)
slice_act_cr_qp_offset	se(v)
}	
if(chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)

if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag || slice_sao_chroma_flag || !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	
if(tiles_enabled_flag || entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
for(i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[i]	u(8)
}	
byte_alignment( )	
}	

7.3.6.2 Reference picture list modification syntax

ref_pic_lists_modification( ) {	**Descriptor**
ref_pic_list_modification_flag_l0	u(1)
if(ref_pic_list_modification_flag_l0)	
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
list_entry_l0[i]	u(v)
if(slice_type = = B) {	
ref_pic_list_modification_flag_l1	u(1)
if(ref_pic_list_modification_flag_l1)	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
list_entry_l1[i]	u(v)
}	
}	

7.3.6.3 Weighted prediction parameters syntax

pred_weight_table( ) {	**Descriptor**
luma_log2_weight_denom	ue(v)
if(ChromaArrayType != 0)	
delta_chroma_log2_weight_denom	se(v)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) || (PicOrderCnt(RefPicList0[i]) != PicOrderCnt(CurrPic)))	
luma_weight_l0_flag[i]	u(1)
if(ChromaArrayType != 0)	
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) || (PicOrderCnt(RefPicList0[i]) != PicOrderCnt(CurrPic)))	
chroma_weight_l0_flag[i]	u(1)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++) {	
if(luma_weight_l0_flag[i]) {	
delta_luma_weight_l0[i]	se(v)
luma_offset_l0[i]	se(v)
}	
if(chroma_weight_l0_flag[i])	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_l0[i][j]	se(v)
delta_chroma_offset_l0[i][j]	se(v)
}	
}	
if(slice_type = = B) {	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) || (PicOrderCnt(RefPicList1[i]) != PicOrderCnt(CurrPic)))	
luma_weight_l1_flag[i]	u(1)
if(ChromaArrayType != 0)	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) || (PicOrderCnt(RefPicList1[i]) != PicOrderCnt(CurrPic)))	
chroma_weight_l1_flag[i]	u(1)
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++) {	
if(luma_weight_l1_flag[i]) {	
delta_luma_weight_l1[i]	se(v)
luma_offset_l1[i]	se(v)
}	

if(chroma_weight_l1_flag[i])	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_l1[i][j]	se(v)
delta_chroma_offset_l1[i][j]	se(v)
}	
}	
}	
}	

7.3.7 Short-term reference picture set syntax

st_ref_pic_set(stRpsIdx) {	**Descriptor**
if(stRpsIdx != 0)	
inter_ref_pic_set_prediction_flag	u(1)
if(inter_ref_pic_set_prediction_flag) {	
if(stRpsIdx = = num_short_term_ref_pic_sets)	
delta_idx_minus1	ue(v)
delta_rps_sign	u(1)
abs_delta_rps_minus1	ue(v)
for(j = 0; j <= NumDeltaPocs[RefRpsIdx]; j++) {	
used_by_curr_pic_flag[j]	u(1)
if(!used_by_curr_pic_flag[j])	
use_delta_flag[j]	u(1)
}	
} else {	
num_negative_pics	ue(v)
num_positive_pics	ue(v)
for(i = 0; i < num_negative_pics; i++) {	
delta_poc_s0_minus1[i]	ue(v)
used_by_curr_pic_s0_flag[i]	u(1)
}	
for(i = 0; i < num_positive_pics; i++) {	
delta_poc_s1_minus1[i]	ue(v)
used_by_curr_pic_s1_flag[i]	u(1)
}	
}	
}	

7.3.8 Slice segment data syntax

7.3.8.1 General slice segment data syntax

slice_segment_data( ) {	**Descriptor**
do {	
coding_tree_unit( )	
end_of_slice_segment_flag	ae(v)
CtbAddrInTs++	
CtbAddrInRs = CtbAddrTsToRs[CtbAddrInTs]	
if(!end_of_slice_segment_flag && ((tiles_enabled_flag && TileId[CtbAddrInTs] != TileId[CtbAddrInTs − 1]) || (entropy_coding_sync_enabled_flag && (CtbAddrInRs % PicWidthInCtbsY = = 0 || TileId[CtbAddrInTs] != TileId[CtbAddrRsToTs[CtbAddrInRs − 1]])))) {	
end_of_subset_one_bit /* equal to 1 */	ae(v)
byte_alignment( )	
}	
} while(!end_of_slice_segment_flag)	
}	

7.3.8.2 Coding tree unit syntax

coding_tree_unit( ) {	**Descriptor**
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
if(slice_sao_luma_flag || slice_sao_chroma_flag)	
sao(xCtb >> CtbLog2SizeY, yCtb >> CtbLog2SizeY)	
coding_quadtree(xCtb, yCtb, CtbLog2SizeY, 0)	
}	

7.3.8.3 Sample adaptive offset syntax

sao(rx, ry) {	**Descriptor**
if(rx > 0) {	
leftCtbInSliceSeg = CtbAddrInRs > SliceAddrRs	
leftCtbInTile = TileId[CtbAddrInTs] = = TileId[CtbAddrRsToTs[CtbAddrInRs − 1]]	
if(leftCtbInSliceSeg && leftCtbInTile)	
sao_merge_left_flag	ae(v)
}	
if(ry > 0 && !sao_merge_left_flag) {	
upCtbInSliceSeg = (CtbAddrInRs − PicWidthInCtbsY) >= SliceAddrRs	
upCtbInTile = TileId[CtbAddrInTs] = = TileId[CtbAddrRsToTs[CtbAddrInRs − PicWidthInCtbsY]]	
if(upCtbInSliceSeg && upCtbInTile)	
sao_merge_up_flag	ae(v)
}	
if(!sao_merge_up_flag && !sao_merge_left_flag)	
for(cIdx = 0; cIdx < (ChromaArrayType != 0 ? 3 : 1); cIdx++)	
if((slice_sao_luma_flag && cIdx = = 0) || (slice_sao_chroma_flag && cIdx > 0)) {	
if(cIdx = = 0)	
sao_type_idx_luma	ae(v)
else if(cIdx = = 1)	
sao_type_idx_chroma	ae(v)
if(SaoTypeIdx[cIdx][rx][ry] != 0) {	
for(i = 0; i < 4; i++)	
sao_offset_abs[cIdx][rx][ry][i]	ae(v)
if(SaoTypeIdx[cIdx][rx][ry] = = 1) {	
for(i = 0; i < 4; i++)	
if(sao_offset_abs[cIdx][rx][ry][i] != 0)	
sao_offset_sign[cIdx][rx][ry][i]	ae(v)
sao_band_position[cIdx][rx][ry]	ae(v)
} else {	
if(cIdx = = 0)	
sao_eo_class_luma	ae(v)
if(cIdx = = 1)	
sao_eo_class_chroma	ae(v)
}	
}	
}	
}	

7.3.8.4 Coding quadtree syntax

coding_quadtree(x0, y0, log2CbSize, cqtDepth) {	**Descriptor**
if(x0 + (1 << log2CbSize) <= pic_width_in_luma_samples && y0 + (1 << log2CbSize) <= pic_height_in_luma_samples && log2CbSize > MinCbLog2SizeY)	
split_cu_flag[x0][y0]	ae(v)
if(cu_qp_delta_enabled_flag && log2CbSize >= Log2MinCuQpDeltaSize) {	
IsCuQpDeltaCoded = 0	
CuQpDeltaVal = 0	
}	
if(cu_chroma_qp_offset_enabled_flag && log2CbSize >= Log2MinCuChromaQpOffsetSize)	
IsCuChromaQpOffsetCoded = 0	
if(split_cu_flag[x0][y0]) {	
x1 = x0 + (1 << (log2CbSize − 1))	
y1 = y0 + (1 << (log2CbSize − 1))	
coding_quadtree(x0, y0, log2CbSize − 1, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples)	
coding_quadtree(x1, y0, log2CbSize − 1, cqtDepth + 1)	
if(y1 < pic_height_in_luma_samples)	
coding_quadtree(x0, y1, log2CbSize − 1, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)	
coding_quadtree(x1, y1, log2CbSize − 1, cqtDepth + 1)	
} else	
coding_unit(x0, y0, log2CbSize)	
}	

7.3.8.5 Coding unit syntax

coding_unit(x0, y0, log2CbSize) {	**Descriptor**
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)
nCbS = (1 << log2CbSize)	
if(cu_skip_flag[x0][y0])	
prediction_unit(x0, y0, nCbS, nCbS)	
else {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(palette_mode_enabled_flag && CuPredMode[x0][y0] = = MODE_INTRA && log2CbSize <= MaxTbLog2SizeY)	
palette_mode_flag[x0][y0]	ae(v)
if(palette_mode_flag[x0][y0])	
palette_coding(x0, y0, nCbS)	
else {	
if(CuPredMode[x0][y0] != MODE_INTRA || log2CbSize = = MinCbLog2SizeY)	

part_mode	ae(v)
if(CuPredMode[x0][y0] = = MODE_INTRA) {	
if(PartMode = = PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	f(1)
pcm_sample(x0, y0, log2CbSize)	
} else {	
pbOffset = (PartMode = = PART_NxN) ? (nCbS / 2) : nCbS	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
prev_intra_luma_pred_flag[x0 + i][y0 + j]	ae(v)
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
if(prev_intra_luma_pred_flag[x0 + i][y0 + j])	
mpm_idx[x0 + i][y0 + j]	ae(v)
else	
rem_intra_luma_pred_mode[x0 + i][y0 + j]	ae(v)
if(ChromaArrayType = = 3)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
intra_chroma_pred_mode[x0 + i][y0 + j]	ae(v)
else if(ChromaArrayType != 0)	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	
} else {	
if(PartMode = = PART_2Nx2N)	
prediction_unit(x0, y0, nCbS, nCbS)	
else if(PartMode = = PART_2NxN) {	
prediction_unit(x0, y0, nCbS, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS, nCbS / 2)	
} else if(PartMode = = PART_Nx2N) {	
prediction_unit(x0, y0, nCbS / 2, nCbS)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS)	
} else if(PartMode = = PART_2NxnU) {	
prediction_unit(x0, y0, nCbS, nCbS / 4)	
prediction_unit(x0, y0 + (nCbS / 4), nCbS, nCbS * 3 / 4)	
} else if(PartMode = = PART_2NxnD) {	
prediction_unit(x0, y0, nCbS, nCbS * 3 / 4)	
prediction_unit(x0, y0 + (nCbS * 3 / 4), nCbS, nCbS / 4)	
} else if(PartMode = = PART_nLx2N) {	
prediction_unit(x0, y0, nCbS / 4, nCbS)	
prediction_unit(x0 + (nCbS / 4), y0, nCbS * 3 / 4, nCbS)	
} else if(PartMode = = PART_nRx2N) {	
prediction_unit(x0, y0, nCbS * 3 / 4, nCbS)	
prediction_unit(x0 + (nCbS * 3 / 4), y0, nCbS / 4, nCbS)	

} else { /* PART_NxN */	
prediction_unit(x0, y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
}	
}	
if(!pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA && !(PartMode = = PART_2Nx2N && merge_flag[x0][y0]))	
rqt_root_cbf	ae(v)
if(rqt_root_cbf) {	
MaxTrafoDepth = (CuPredMode[x0][y0] = = MODE_INTRA ? (max_transform_hierarchy_depth_intra + IntraSplitFlag) : max_transform_hierarchy_depth_inter)	
transform_tree(x0, y0, x0, y0, log2CbSize, 0, 0)	
}	
}	
}	
}	
}	

7.3.8.6 Prediction unit syntax

prediction_unit(x0, y0, nPbW, nPbH) {	**Descriptor**
if(cu_skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
} else { /* MODE_INTER */	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
} else {	
if(slice_type = = B)	
inter_pred_idc[x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(x0, y0, 0)	
mvp_l0_flag[x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)

if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] = = PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
} else	
mvd_coding(x0, y0, 1)	
mvp_l1_flag[x0][y0]	ae(v)
}	
}	
}	
}	

7.3.8.7 PCM sample syntax

pcm_sample(x0, y0, log2CbSize) {	**Descriptor**
for(i = 0; i < 1 << (log2CbSize << 1); i++)	
pcm_sample_luma[i]	u(v)
if(ChromaArrayType != 0)	
for(i = 0; i < ((2 << (log2CbSize << 1)) / (SubWidthC * SubHeightC)); i++)	
pcm_sample_chroma[i]	u(v)
}	

7.3.8.8 Transform tree syntax

transform_tree(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) {	**Descriptor**
if(log2TrafoSize <= MaxTbLog2SizeY && log2TrafoSize > MinTbLog2SizeY && trafoDepth < MaxTrafoDepth && !(IntraSplitFlag && (trafoDepth = = 0)))	
split_transform_flag[x0][y0][trafoDepth]	ae(v)
if((log2TrafoSize > 2 && ChromaArrayType != 0) || ChromaArrayType = = 3) {	
if(trafoDepth = = 0 || cbf_cb[xBase][yBase][trafoDepth − 1]) {	
cbf_cb[x0][y0][trafoDepth]	ae(v)
if(ChromaArrayType = = 2 && (!split_transform_flag[x0][y0][trafoDepth] || log2TrafoSize = = 3))	
cbf_cb[x0][y0 + (1 << (log2TrafoSize − 1))][trafoDepth]	ae(v)
}	
if(trafoDepth = = 0 || cbf_cr[xBase][yBase][trafoDepth − 1]) {	
cbf_cr[x0][y0][trafoDepth]	ae(v)
if(ChromaArrayType = = 2 && (!split_transform_flag[x0][y0][trafoDepth] || log2TrafoSize = = 3))	
cbf_cr[x0][y0 + (1 << (log2TrafoSize − 1))][trafoDepth]	ae(v)
}	
}	
if(split_transform_flag[x0][y0][trafoDepth]) {	
x1 = x0 + (1 << (log2TrafoSize − 1))	
y1 = y0 + (1 << (log2TrafoSize − 1))	

transform_tree(x0, y0, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 0)	
transform_tree(x1, y0, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 1)	
transform_tree(x0, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 2)	
transform_tree(x1, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 3)	
} else {	
if(CuPredMode[x0][y0] = = MODE_INTRA || trafoDepth != 0 || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth] || (ChromaArrayType = = 2 && (cbf_cb[x0][y0 + (1 << (log2TrafoSize − 1))][trafoDepth] || cbf_cr[x0][y0 + (1 << (log2TrafoSize − 1))][trafoDepth])))	
cbf_luma[x0][y0][trafoDepth]	ae(v)
transform_unit(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx)	
}	
}	

7.3.8.9 Motion vector difference syntax

mvd_coding(x0, y0, refList) {	Descriptor
abs_mvd_greater0_flag[0]	ae(v)
abs_mvd_greater0_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0])	
abs_mvd_greater1_flag[0]	ae(v)
if(abs_mvd_greater0_flag[1])	
abs_mvd_greater1_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0]) {	
if(abs_mvd_greater1_flag[0])	
abs_mvd_minus2[0]	ae(v)
mvd_sign_flag[0]	ae(v)
}	
if(abs_mvd_greater0_flag[1]) {	
if(abs_mvd_greater1_flag[1])	
abs_mvd_minus2[1]	ae(v)
mvd_sign_flag[1]	ae(v)
}	
}	

7.3.8.10 Transform unit syntax

transform_unit(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) {	Descriptor
log2TrafoSizeC = Max(2, log2TrafoSize − (ChromaArrayType = = 3 ? 0 : 1))	
cbfDepthC = trafoDepth − (ChromaArrayType != 3 && log2TrafoSize = = 2 ? 1 : 0)	
xC = (ChromaArrayType != 3 && log2TrafoSize = = 2) ? xBase : x0	
yC = (ChromaArrayType != 3 && log2TrafoSize = = 2) ? yBase : y0	
cbfLuma = cbf_luma[x0][y0][trafoDepth]	

cbfChroma = cbf_cb[xC][yC][cbfDepthC] || cbf_cr[xC][yC][cbfDepthC] || (ChromaArrayType = = 2 && (cbf_cb[xC][yC + (1 << log2TrafoSizeC)][cbfDepthC] || cbf_cr[xC][yC + (1 << log2TrafoSizeC)][cbfDepthC]))	
if(cbfLuma || cbfChroma) {	
xP = (x0 >> MinCbLog2SizeY) << MinCbLog2SizeY	
yP = (y0 >> MinCbLog2SizeY) << MinCbLog2SizeY	
nCbS = 1 << MinCbLog2SizeY	
if(residual_adaptive_colour_transform_enabled_flag && (CuPredMode[x0][y0] = = MODE_INTER || (PartMode = = PART_2Nx2N && intra_chroma_pred_mode[x0][y0] = = 4) || (intra_chroma_pred_mode[xP][yP] = = 4 && intra_chroma_pred_mode[xP + nCbS/2][yP] = = 4 && intra_chroma_pred_mode[xP][yP + nCbS/2] = = 4 && intra_chroma_pred_mode[xP + nCbS/2][yP + nCbS/2] = = 4)))	
tu_residual_act_flag[x0][y0]	ae(v)
delta_qp( )	
if(cbfChroma && !cu_transquant_bypass_flag)	
chroma_qp_offset( )	
if(cbfLuma)	
residual_coding(x0, y0, log2TrafoSize, 0)	
if(log2TrafoSize > 2 || ChromaArrayType = = 3) {	
if(cross_component_prediction_enabled_flag && cbfLuma && (CuPredMode[x0][y0] = = MODE_INTER || intra_chroma_pred_mode[x0][y0] = = 4))	
cross_comp_pred(x0, y0, 0)	
for(tIdx = 0; tIdx < (ChromaArrayType = = 2 ? 2 : 1); tIdx++)	
if(cbf_cb[x0][y0 + (tIdx << log2TrafoSizeC)][trafoDepth])	
residual_coding(x0, y0 + (tIdx << log2TrafoSizeC), log2TrafoSizeC, 1)	
if(cross_component_prediction_enabled_flag && cbfLuma && (CuPredMode[x0][y0] = = MODE_INTER || intra_chroma_pred_mode[x0][y0] = = 4))	
cross_comp_pred(x0, y0, 1)	
for(tIdx = 0; tIdx < (ChromaArrayType = = 2 ? 2 : 1); tIdx++)	
if(cbf_cr[x0][y0 + (tIdx << log2TrafoSizeC)][trafoDepth])	
residual_coding(x0, y0 + (tIdx << log2TrafoSizeC), log2TrafoSizeC, 2)	
} else if(blkIdx = = 3) {	
for(tIdx = 0; tIdx < (ChromaArrayType = = 2 ? 2 : 1); tIdx++)	
if(cbf_cb[xBase][yBase + (tIdx << log2TrafoSizeC)][trafoDepth − 1])	
residual_coding(xBase, yBase + (tIdx << log2TrafoSizeC), log2TrafoSize, 1)	
for(tIdx = 0; tIdx < (ChromaArrayType = = 2 ? 2 : 1); tIdx++)	
if(cbf_cr[xBase][yBase + (tIdx << log2TrafoSizeC)][trafoDepth − 1])	
residual_coding(xBase, yBase + (tIdx << log2TrafoSizeC), log2TrafoSize, 2)	
}	
}	
}	

7.3.8.11 Residual coding syntax

residual_coding(x0, y0, log2TrafoSize, cIdx) {	**Descriptor**
if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize <= Log2MaxTransformSkipSize))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
if(CuPredMode[x0][y0] = = MODE_INTER && explicit_rdpcm_enabled_flag && (transform_skip_flag[x0][y0][cIdx] | | cu_transquant_bypass_flag)) {	
explicit_rdpcm_flag[x0][y0][cIdx]	ae(v)
if(explicit_rdpcm_flag[x0][y0][cIdx])	
explicit_rdpcm_dir_flag[x0][y0][cIdx]	ae(v)
}	
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
lastScanPos = 16	
lastSubBlock = (1 << (log2TrafoSize − 2)) * (1 << (log2TrafoSize − 2)) − 1	
do {	
if(lastScanPos = = 0) {	
lastScanPos = 16	
lastSubBlock− −	
}	
lastScanPos− −	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) | | (yC != LastSignificantCoeffY))	
for(i = lastSubBlock; i >= 0; i− −) {	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][i][1]	
escapeDataPresent = 0	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
for(n = (i = = lastSubBlock) ? lastScanPos − 1 : 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 | | !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
}	

firstSigScanPos = 16	
lastSigScanPos = −1	
numGreater1Flag = 0	
lastGreater1ScanPos = −1	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
if(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
if(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos = = −1)	
lastGreater1ScanPos = n	
else if(coeff_abs_level_greater1_flag[n])	
escapeDataPresent = 1	
} else	
escapeDataPresent = 1	
if(lastSigScanPos = = −1)	
lastSigScanPos = n	
firstSigScanPos = n	
}	
}	
if(cu_transquant_bypass_flag || (CuPredMode[x0][y0] = = MODE_INTRA && implicit_rdpcm_enabled_flag && transform_skip_flag[x0][y0][cIdx] && (predModeIntra = = 10 || predModeIntra = = 26)) || explicit_rdpcm_flag[x0][y0][cIdx])	
signHidden = 0	
else	
signHidden = lastSigScanPos − firstSigScanPos > 3	
if(lastGreater1ScanPos != −1) {	
coeff_abs_level_greater2_flag[lastGreater1ScanPos]	ae(v)
if(coeff_abs_level_greater2_flag[lastGreater1ScanPos])	
escapeDataPresent = 1	
}	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC] && (!sign_data_hiding_enabled_flag || !signHidden || (n != firstSigScanPos)))	
coeff_sign_flag[n]	ae(v)
}	
numSigCoeff = 0	
sumAbsLevel = 0	
for(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	

baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
if(baseLevel = = ((numSigCoeff < 8) ? ((n = = lastGreater1ScanPos) ? 3 : 2) : 1))	
coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 − 2 * coeff_sign_flag[n])	
if(sign_data_hiding_enabled_flag && signHidden) {	
sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)	
if((n = = firstSigScanPos) && ((sumAbsLevel % 2) = = 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
numSigCoeff++	
}	
}	
}	
}	

7.3.8.12 Cross-component prediction syntax

cross_comp_pred(x0, y0, c) {	**Descriptor**
log2_res_scale_abs_plus1[c]	ae(v)
if(log2_res_scale_abs_plus1[c] != 0)	
res_scale_sign_flag[c]	ae(v)
}	

7.3.8.13 Palette syntax

palette_coding(x0, y0, nCbS) {	**Descriptor**
palettePredictionFinished = 0	
NumPredictedPaletteEntries = 0	
for(predictorEntryIdx = 0; predictorEntryIdx < PredictorPaletteSize && !palettePredictionFinished && NumPredictedPaletteEntries < palette_max_size; predictorEntryIdx++) {	
palette_predictor_run	ae(v)
if(palette_predictor_run != 1) {	
if(palette_predictor_run > 1)	
predictorEntryIdx += palette_predictor_run − 1	
PalettePredictorEntryReuseFlags[predictorEntryIdx] = 1	
NumPredictedPaletteEntries++	
} else	
palettePredictionFinished = 1	
}	
if(NumPredictedPaletteEntries < palette_max_size)	
num_signalled_palette_entries	ae(v)
numComps = (ChromaArrayType = = 0) ? 1 : 3	

for(cIdx = 0; cIdx < numComps; cIdx++)	
for(i = 0; i < num_signalled_palette_entries; i++)	
new_palette_entries[cIdx][i]	ae(v)
if(CurrentPaletteSize != 0)	
palette_escape_val_present_flag	ae(v)
if(MaxPaletteIndex > 0) {	
num_palette_indices_minus1	ae(v)
adjust = 0	
for(i = 0; i <= num_palette_indices_minus1; i++) {	
if(MaxPaletteIndex − adjust > 0) {	
palette_idx_idc	ae(v)
PaletteIndexIdc[i] = palette_idx_idc	
}	
adjust = 1	
}	
copy_above_indices_for_final_run_flag	ae(v)
palette_transpose_flag	ae(v)
}	
if(palette_escape_val_present_flag) {	
delta_qp( )	
if(!cu_transquant_bypass_flag)	
chroma_qp_offset( )	
}	
remainingNumIndices = num_palette_indices_minus1 + 1	
PaletteScanPos = 0	
log2BlockSize = Log2(nCbS)	
while(PaletteScanPos < nCbS * nCbS) {	
xC = x0 + ScanOrder[log2BlockSize][3][PaletteScanPos][0]	
yC = y0 + ScanOrder[log2BlockSize][3][PaletteScanPos][1]	
if(PaletteScanPos > 0) {	
xcPrev = x0 + ScanOrder[log2BlockSize][3][PaletteScanPos − 1][0]	
ycPrev = y0 + ScanOrder[log2BlockSize][3][PaletteScanPos − 1][1]	
}	
PaletteRunMinus1 = nCbS * nCbS − PaletteScanPos − 1	
RunToEnd = 1	
CopyAboveIndicesFlag[xC][yC] = 0	
if(MaxPaletteIndex > 0)	
if(PaletteScanPos >= nCbS && CopyAboveIndicesFlag[xcPrev][ycPrev] = = 0)	
if(remainingNumIndices > 0 && PaletteScanPos < nCbS * nCbS − 1) {	
copy_above_palette_indices_flag	ae(v)
CopyAboveIndicesFlag[xC][yC] = copy_above_palette_indices_flag	
} else	
if(PaletteScanPos = = nCbS * nCbS − 1 && remainingNumIndices > 0)	
CopyAboveIndicesFlag[xC][yC] = 0	
else	
CopyAboveIndicesFlag[xC][yC] = 1	

if(CopyAboveIndicesFlag[xC][yC] = = 0) {	
currNumIndices = num_palette_indices_minus1 + 1 − remainingNumIndices	
CurrPaletteIndex = PaletteIndexIdc[currNumIndices]	
}	
if(MaxPaletteIndex > 0) {	
if(CopyAboveIndicesFlag[xC][yC] = = 0)	
remainingNumIndices − = 1	
if(remainingNumIndices > 0 || CopyAboveIndicesFlag[xC][yC] != copy_above_indices_for_final_run_flag) {	
PaletteMaxRunMinus1 = nCbS * nCbS − PaletteScanPos − 1 − remainingNumIndices − copy_above_indices_for_final_run_flag	
RunToEnd = 0	
if(PaletteMaxRunMinus1 > 0) {	
palette_run_prefix	ae(v)
if((palette_run_prefix > 1) && (PaletteMaxRunMinus1 != (1 << (palette_run_prefix − 1))))	
palette_run_suffix	ae(v)
}	
}	
}	
runPos = 0	
while (runPos <= PaletteRunMinus1) {	
xR = x0 + ScanOrder[log2BlockSize][3][PaletteScanPos][0]	
yR = y0 + ScanOrder[log2BlockSize][3][PaletteScanPos][1]	
if(CopyAboveIndicesFlag[xC][yC] = = 0) {	
CopyAboveIndicesFlag[xR][yR] = 0	
PaletteIndexMap[xR][yR] = CurrPaletteIndex	
} else {	
CopyAboveIndicesFlag[xR][yR] = 1	
PaletteIndexMap[xR][yR] = PaletteIndexMap[xR][yR − 1]	
}	
runPos++	
PaletteScanPos++	
}	
}	
if(palette_escape_val_present_flag) {	
for(cIdx = 0; cIdx < numComps; cIdx++)	
for(sPos = 0; sPos < nCbS * nCbS; sPos++) {	
xC = x0 + ScanOrder[log2BlockSize][3][sPos][0]	
yC = y0 + ScanOrder[log2BlockSize][3][sPos][1]	
if(PaletteIndexMap[xC][yC] = = MaxPaletteIndex)	
if(cIdx = = 0 || (xC % 2 = = 0 && yC % 2 = = 0 && ChromaArrayType = = 1) || (xC % 2 = = 0 && !palette_transpose_flag && ChromaArrayType = = 2) || (yC % 2 = = 0 && palette_transpose_flag && ChromaArrayType = = 2) || ChromaArrayType = = 3) {	

palette_escape_val	ae(v)
PaletteEscapeVal[cIdx][xC][yC] = palette_escape_val	
}	
}	
}	
}	

7.3.8.14 Delta QP syntax

delta_qp( ) {	**Descriptor**
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
IsCuQpDeltaCoded = 1	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs) {	
cu_qp_delta_sign_flag	ae(v)
CuQpDeltaVal = cu_qp_delta_abs * (1 − 2 * cu_qp_delta_sign_flag)	
}	
}	
}	

7.3.8.15 Chroma QP offset syntax

chroma_qp_offset( ) {	**Descriptor**
if(cu_chroma_qp_offset_enabled_flag && !IsCuChromaQpOffsetCoded) {	
cu_chroma_qp_offset_flag	ae(v)
if(cu_chroma_qp_offset_flag && chroma_qp_offset_list_len_minus1 > 0)	
cu_chroma_qp_offset_idx	ae(v)
}	
}	

7.4 Semantics

7.4.1 General

Semantics associated with the syntax structures and with the syntax elements within these structures are specified in this clause. When the semantics of a syntax element are specified using a table or a set of tables, any values that are not specified in the table(s) shall not be present in the bitstream unless otherwise specified in this Specification.

7.4.2 NAL unit semantics

7.4.2.1 General NAL unit semantics

NumBytesInNalUnit specifies the size of the NAL unit in bytes. This value is required for decoding of the NAL unit. Some form of demarcation of NAL unit boundaries is necessary to enable inference of NumBytesInNalUnit. One such demarcation method is specified in Annex B for the byte stream format. Other methods of demarcation may be specified outside of this Specification.

NOTE 1 – The video coding layer (VCL) is specified to efficiently represent the content of the video data. The NAL is specified to format that data and provide header information in a manner appropriate for conveyance on a variety of communication channels or storage media. All data are contained in NAL units, each of which contains an integer number of bytes. A NAL unit specifies a generic format for use in both packet-oriented and bitstream systems. The format of NAL units for both packet-oriented transport and byte stream is identical except that each NAL unit can be preceded by a start code prefix and extra padding bytes in the byte stream format specified in Annex B.

rbsp_byte[i] is the i-th byte of an RBSP. An RBSP is specified as an ordered sequence of bytes as follows:

The RBSP contains a string of data bits (SODB) as follows:

– If the SODB is empty (i.e., zero bits in length), the RBSP is also empty.
– Otherwise, the RBSP contains the SODB as follows:
  1) The first byte of the RBSP contains the first (most significant, left-most) eight bits of the SODB; the next byte of the RBSP contains the next eight bits of the SODB, etc., until fewer than eight bits of the SODB remain.
  2) The rbsp_trailing_bits( ) syntax structure is present after the SODB as follows:
     i) The first (most significant, left-most) bits of the final RBSP byte contain the remaining bits of the SODB (if any).
     ii) The next bit consists of a single bit equal to 1 (i.e., rbsp_stop_one_bit).
     iii) When the rbsp_stop_one_bit is not the last bit of a byte-aligned byte, one or more zero-valued bits (i.e., instances of rbsp_alignment_zero_bit) are present to result in byte alignment.
  3) One or more cabac_zero_word 16-bit syntax elements equal to 0x0000 may be present in some RBSPs after the rbsp_trailing_bits( ) at the end of the RBSP.

Syntax structures having these RBSP properties are denoted in the syntax tables using an "_rbsp" suffix. These structures are carried within NAL units as the content of the rbsp_byte[i] data bytes. The association of the RBSP syntax structures to the NAL units is as specified in Table 7-1.

NOTE 2 – When the boundaries of the RBSP are known, the decoder can extract the SODB from the RBSP by concatenating the bits of the bytes of the RBSP and discarding the rbsp_stop_one_bit, which is the last (least significant, right-most) bit equal to 1, and discarding any following (less significant, farther to the right) bits that follow it, which are equal to 0. The data necessary for the decoding process is contained in the SODB part of the RBSP.

emulation_prevention_three_byte is a byte equal to 0x03. When an emulation_prevention_three_byte is present in the NAL unit, it shall be discarded by the decoding process.

The last byte of the NAL unit shall not be equal to 0x00.

Within the NAL unit, the following three-byte sequences shall not occur at any byte-aligned position:

– 0x000000
– 0x000001
– 0x000002

Within the NAL unit, any four-byte sequence that starts with 0x000003 other than the following sequences shall not occur at any byte-aligned position:

– 0x00000300
– 0x00000301
– 0x00000302
– 0x00000303

7.4.2.2 NAL unit header semantics

forbidden_zero_bit shall be equal to 0.

nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table 7-1.

NAL units that have nal_unit_type in the range of UNSPEC48..UNSPEC63, inclusive, for which semantics are not specified, shall not affect the decoding process specified in this Specification.

NOTE 1 – NAL unit types in the range of UNSPEC48..UNSPEC63 may be used as determined by the application. No decoding process for these values of nal_unit_type is specified in this Specification. Since different applications might use these NAL unit types for different purposes, particular care must be exercised in the design of encoders that generate NAL units with these nal_unit_type values, and in the design of decoders that interpret the content of NAL units with these nal_unit_type values. This Specification does not define any management for these values. These nal_unit_type values might only be suitable for use in contexts in which "collisions" of usage (i.e., different definitions of the meaning of the NAL unit content for the same nal_unit_type value) are unimportant, or not possible, or are managed – e.g., defined or managed in the controlling application or transport specification, or by controlling the environment in which bitstreams are distributed.

For purposes other than determining the amount of data in the decoding units of the bitstream (as specified in Annex C), decoders shall ignore (remove from the bitstream and discard) the contents of all NAL units that use reserved values of nal_unit_type.

NOTE 2 – This requirement allows future definition of compatible extensions to this Specification.

Table 7-1 – NAL unit type codes and NAL unit type classes

nal_unit_type	Name of nal_unit_type	Content of NAL unit and RBSP syntax structure	NAL unit type class
0 1	TRAIL_N TRAIL_R	Coded slice segment of a non-TSA, non-STSA trailing picture slice_segment_layer_rbsp( )	VCL
2 3	TSA_N TSA_R	Coded slice segment of a TSA picture slice_segment_layer_rbsp( )	VCL
4 5	STSA_N STSA_R	Coded slice segment of an STSA picture slice_segment_layer_rbsp( )	VCL
6 7	RADL_N RADL_R	Coded slice segment of a RADL picture slice_segment_layer_rbsp( )	VCL
8 9	RASL_N RASL_R	Coded slice segment of a RASL picture slice_segment_layer_rbsp( )	VCL
10 12 14	RSV_VCL_N10 RSV_VCL_N12 RSV_VCL_N14	Reserved non-IRAP SLNR VCL NAL unit types	VCL
11 13 15	RSV_VCL_R11 RSV_VCL_R13 RSV_VCL_R15	Reserved non-IRAP sub-layer reference VCL NAL unit types	VCL
16 17 18	BLA_W_LP BLA_W_RADL BLA_N_LP	Coded slice segment of a BLA picture slice_segment_layer_rbsp( )	VCL
19 20	IDR_W_RADL IDR_N_LP	Coded slice segment of an IDR picture slice_segment_layer_rbsp( )	VCL
21	CRA_NUT	Coded slice segment of a CRA picture slice_segment_layer_rbsp( )	VCL
22 23	RSV_IRAP_VCL22 RSV_IRAP_VCL23	Reserved IRAP VCL NAL unit types	VCL
24..31	RSV_VCL24.. RSV_VCL31	Reserved non-IRAP VCL NAL unit types	VCL
32	VPS_NUT	Video parameter set video_parameter_set_rbsp( )	non-VCL
33	SPS_NUT	Sequence parameter set seq_parameter_set_rbsp( )	non-VCL
34	PPS_NUT	Picture parameter set pic_parameter_set_rbsp( )	non-VCL
35	AUD_NUT	Access unit delimiter access_unit_delimiter_rbsp( )	non-VCL
36	EOS_NUT	End of sequence end_of_seq_rbsp( )	non-VCL
37	EOB_NUT	End of bitstream end_of_bitstream_rbsp( )	non-VCL
38	FD_NUT	Filler data filler_data_rbsp( )	non-VCL
39 40	PREFIX_SEI_NUT SUFFIX_SEI_NUT	Supplemental enhancement information sei_rbsp( )	non-VCL
41..47	RSV_NVCL41.. RSV_NVCL47	Reserved	non-VCL
48..63	UNSPEC48.. UNSPEC63	Unspecified	non-VCL

NOTE 3 – A clean random access (CRA) picture may have associated random access skipped leading (RASL) or random access decodable leading (RADL) pictures present in the bitstream.

NOTE 4 – A broken link access (BLA) picture having nal_unit_type equal to BLA_W_LP may have associated RASL or RADL pictures present in the bitstream. A BLA picture having nal_unit_type equal to BLA_W_RADL does not have associated RASL pictures present in the bitstream, but may have associated RADL pictures in the bitstream. A BLA picture having nal_unit_type equal to BLA_N_LP does not have associated leading pictures present in the bitstream.

NOTE 5 – An instantaneous decoding refresh (IDR) picture having nal_unit_type equal to IDR_N_LP does not have associated leading pictures present in the bitstream. An IDR picture having nal_unit_type equal to IDR_W_RADL does not have associated RASL pictures present in the bitstream, but may have associated RADL pictures in the bitstream.

NOTE 6 – A sub-layer non-reference (SLNR) picture is not included in any of RefPicSetStCurrBefore, RefPicSetStCurrAfter and RefPicSetLtCurr of any picture with the same value of TemporalId, and may be discarded without affecting the decodability of other pictures with the same value of TemporalId.

All coded slice segment NAL units of an access unit shall have the same value of nal_unit_type. A picture or an access unit is also referred to as having a nal_unit_type equal to the nal_unit_type of the coded slice segment NAL units of the picture or the access unit.

If a picture has nal_unit_type equal to TRAIL_N, TSA_N, STSA_N, RADL_N, RASL_N, RSV_VCL_N10, RSV_VCL_N12 or RSV_VCL_N14, the picture is an SLNR picture. Otherwise, the picture is a sub-layer reference picture.

Each picture, other than the first picture in the bitstream in decoding order, is considered to be associated with the previous intra random access point (IRAP) picture in decoding order.

When a picture is a leading picture, it shall be a RADL or RASL picture.

When a picture is a trailing picture, it shall not be a RADL or RASL picture.

When a picture is a leading picture, it shall precede, in decoding order, all trailing pictures that are associated with the same IRAP picture.

No RASL pictures shall be present in the bitstream that are associated with a BLA picture having nal_unit_type equal to BLA_W_RADL or BLA_N_LP.

No RASL pictures shall be present in the bitstream that are associated with an IDR picture.

No RADL pictures shall be present in the bitstream that are associated with a BLA picture having nal_unit_type equal to BLA_N_LP or that are associated with an IDR picture having nal_unit_type equal to IDR_N_LP.

NOTE 7 – It is possible to perform random access at the position of an IRAP access unit by discarding all access units before the IRAP access unit (and to correctly decode the IRAP picture and all the subsequent non-RASL pictures in decoding order), provided each parameter set is available (either in the bitstream or by external means not specified in this Specification) when it needs to be activated.

Any picture that has PicOutputFlag equal to 1 that precedes an IRAP picture in decoding order shall precede the IRAP picture in output order and shall precede any RADL picture associated with the IRAP picture in output order.

Any RASL picture associated with a CRA or BLA picture shall precede any RADL picture associated with the CRA or BLA picture in output order.

Any RASL picture associated with a CRA picture shall follow, in output order, any IRAP picture that precedes the CRA picture in decoding order.

When sps_temporal_id_nesting_flag is equal to 1 and TemporalId is greater than 0, the nal_unit_type shall be equal to TSA_R, TSA_N, RADL_R, RADL_N, RASL_R or RASL_N.

nuh_layer_id specifies the identifier of the layer to which a VCL NAL unit belongs or the identifier of a layer to which a non-VCL NAL unit applies. The value of nuh_layer_id shall be in the range of 0 to 62, inclusive. The value of 63 may be specified in the future by ITU-T | ISO/IEC. For purposes other than determining the amount of data in the decoding units of the bitstream, decoders shall ignore all data that follow the value 63 for nuh_layer_id in a NAL unit, and decoders conforming to a profile specified in Annex A and not supporting the independent non-base layer decoding (INBLD) capability specified in Annex F shall ignore (i.e., remove from the bitstream and discard) all NAL units with values of nuh_layer_id not equal to 0.

NOTE 8 – The value of 63 for nuh_layer_id may be used to indicate an extended layer identifier in a future extension of this Specification.

The value of nuh_layer_id shall be the same for all VCL NAL units of a coded picture. The value of nuh_layer_id of a coded picture is the value of the nuh_layer_id of the VCL NAL units of the coded picture.

When nal_unit_type is equal to EOB_NUT, the value of nuh_layer_id shall be equal to 0.

nuh_temporal_id_plus1 minus 1 specifies a temporal identifier for the NAL unit. The value of nuh_temporal_id_plus1 shall not be equal to 0.

The variable TemporalId is specified as follows:

$$\text{TemporalId} = \text{nuh\_temporal\_id\_plus1} - 1 \tag{7-1}$$

When nal_unit_type is in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e., the coded slice segment belongs to an IRAP picture, TemporalId shall be equal to 0.

When nal_unit_type is equal to TSA_R or TSA_N, TemporalId shall not be equal to 0.

When nuh_layer_id is equal to 0 and nal_unit_type is equal to STSA_R or STSA_N, TemporalId shall not be equal to 0.

The value of TemporalId shall be the same for all VCL NAL units of an access unit. The value of TemporalId of a coded picture or an access unit is the value of the TemporalId of the VCL NAL units of the coded picture or the access unit. The value of TemporalId of a sub-layer representation is the greatest value of TemporalId of all VCL NAL units in the sub-layer representation.

The value of TemporalId for non-VCL NAL units is constrained as follows:

– If nal_unit_type is equal to VPS_NUT or SPS_NUT, TemporalId shall be equal to 0 and the TemporalId of the access unit containing the NAL unit shall be equal to 0.
– Otherwise if nal_unit_type is equal to EOS_NUT or EOB_NUT, TemporalId shall be equal to 0.
– Otherwise, if nal_unit_type is equal to AUD_NUT or FD_NUT, TemporalId shall be equal to the TemporalId of the access unit containing the NAL unit.
– Otherwise, TemporalId shall be greater than or equal to the TemporalId of the access unit containing the NAL unit.

NOTE 9 – When the NAL unit is a non-VCL NAL unit, the value of TemporalId is equal to the minimum value of the TemporalId values of all access units to which the non-VCL NAL unit applies. When nal_unit_type is equal to PPS_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as all picture parameter sets (PPSs) may be included in the beginning of a bitstream, wherein the first coded picture has TemporalId equal to 0. When nal_unit_type is equal to PREFIX_SEI_NUT or SUFFIX_SEI_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as an SEI NAL unit may contain information, e.g., in a buffering period SEI message or a picture timing SEI message, that applies to a bitstream subset that includes access units for which the TemporalId values are greater than the TemporalId of the access unit containing the SEI NAL unit.

7.4.2.3 Encapsulation of an SODB within an RBSP (informative)

This clause does not form an integral part of this Specification.

The form of encapsulation of an SODB within an RBSP and the use of the emulation_prevention_three_byte for encapsulation of an RBSP within a NAL unit is described for the following purposes:

– To prevent the emulation of start codes within NAL units while allowing any arbitrary SODB to be represented within a NAL unit,
– To enable identification of the end of the SODB within the NAL unit by searching the RBSP for the rbsp_stop_one_bit starting at the end of the RBSP,
– To enable a NAL unit to have a size greater than that of the SODB under some circumstances (using one or more cabac_zero_word syntax elements).

The encoder can produce a NAL unit from an RBSP by the following procedure:

1. The RBSP data are searched for byte-aligned bits of the following binary patterns:

   '00000000 00000000 000000xx' (where 'xx' represents any two-bit pattern: '00', '01', '10', or '11'),

   and a byte equal to 0x03 is inserted to replace the bit pattern with the pattern:

   '00000000 00000000 00000011 000000xx',

   and finally, when the last byte of the RBSP data is equal to 0x00 (which can only occur when the RBSP ends in a cabac_zero_word), a final byte equal to 0x03 is appended to the end of the data. The last zero byte of a byte-aligned three-byte sequence 0x000000 in the RBSP (which is replaced by the four-byte sequence 0x00000300) is taken into account when searching the RBSP data for the next occurrence of byte-aligned bits with the binary patterns specified above.
2. The resulting sequence of bytes is then prefixed with the NAL unit header, within which the nal_unit_type indicates the type of RBSP data structure in the NAL unit.

The process specified above results in the construction of the entire NAL unit.

This process can allow any SODB to be represented in a NAL unit while ensuring both of the following:

– No byte-aligned start code prefix is emulated within the NAL unit.
– No sequence of 8 zero-valued bits followed by a start code prefix, regardless of byte-alignment, is emulated within the NAL unit.

7.4.2.4 Order of NAL units and association to coded pictures, access units and coded video sequences

7.4.2.4.1 General

This clause specifies constraints on the order of NAL units in the bitstream.

Any order of NAL units in the bitstream obeying these constraints is referred to in the text as the decoding order of NAL units. Within a NAL unit, the syntax in clauses 7.3, D.2 and E.2 specifies the decoding order of syntax elements. Decoders shall be capable of receiving NAL units and their syntax elements in decoding order.

7.4.2.4.2 Order of VPS, SPS and PPS RBSPs and their activation

This clause specifies the activation process of video parameter sets (VPSs), sequence parameter sets (SPSs) and PPSs.

NOTE 1 – The VPS, SPS and PPS mechanism decouples the transmission of infrequently changing information from the transmission of coded block data. VPSs, SPSs and PPSs may, in some applications, be conveyed "out-of-band".

A PPS RBSP includes parameters that can be referred to by the coded slice segment NAL units of one or more coded pictures. Each PPS RBSP is initially considered not active for the base layer at the start of the operation of the decoding process. At most one PPS RBSP is considered active for the base layer at any given moment during the operation of the decoding process, and the activation of any particular PPS RBSP for the base layer results in the deactivation of the previously-active PPS RBSP for the base layer (if any).

When a PPS RBSP (with a particular value of pps_pic_parameter_set_id) is not active for the base layer and it is referred to by a coded slice segment NAL unit with nuh_layer_id equal to 0 (using a value of slice_pic_parameter_set_id equal to the pps_pic_parameter_set_id value), it is activated for the base layer. This PPS RBSP is called the active PPS RBSP for the base layer until it is deactivated by the activation of another PPS RBSP for the base layer. A PPS RBSP, with that particular value of pps_pic_parameter_set_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId less than or equal to the TemporalId of the PPS NAL unit or provided through external means, and the PPS NAL unit containing the PPS RBSP shall have nuh_layer_id equal to 0.

Any PPS NAL unit containing the value of pps_pic_parameter_set_id for the active PPS RBSP for a coded picture (and consequently for the layer containing the coded picture) shall have the same content as that of the active PPS RBSP for the coded picture, unless it follows the last VCL NAL unit of the coded picture and precedes the first VCL NAL unit of another coded picture.

An SPS RBSP includes parameters that can be referred to by one or more PPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each SPS RBSP is initially considered not active for the base layer at the start of the operation of the decoding process. At most one SPS RBSP is considered active for the base layer at any given moment during the operation of the decoding process, and the activation of any particular SPS RBSP for the base layer results in the deactivation of the previously-active SPS RBSP for the base layer (if any).

When an SPS RBSP (with a particular value of sps_seq_parameter_set_id) is not already active for the base layer and it is referred to by activation of a PPS RBSP (in which pps_seq_parameter_set_id is equal to the sps_seq_parameter_set_id value) for the base layer or, when vps_base_layer_internal_flag is equal to 1 and vps_base_layer_available_flag is equal to 1, is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which active_seq_parameter_set_id[0] is equal to the sps_seq_parameter_set_id value), it is activated for the base layer. This SPS RBSP is called the active SPS RBSP for the base layer until it is deactivated by the activation of another SPS RBSP for the base layer. An SPS RBSP, with that particular value of sps_seq_parameter_set_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId equal to 0 or provided through external means, and the SPS NAL unit containing the SPS RBSP shall have nuh_layer_id equal to 0. An activated SPS RBSP for the base layer shall remain active for the entire coded video sequence (CVS).

NOTE 2 – Because an IRAP access unit with NoRaslOutputFlag equal to 1 begins a new CVS and an activated SPS RBSP for the base layer must remain active for the entire CVS, an SPS RBSP can only be activated for the base layer by an active parameter sets SEI message when the active parameter sets SEI message is part of an IRAP access unit with NoRaslOutputFlag equal to 1.

Any SPS NAL unit with nuh_layer_id equal to 0 containing the value of sps_seq_parameter_set_id for the active SPS RBSP for the base layer for a CVS shall have the same content as that of the active SPS RBSP for the base layer for the CVS, unless it follows the last access unit of the CVS and precedes the first VCL NAL unit and the first SEI NAL unit containing an active parameter sets SEI message (when present) of another CVS.

A VPS RBSP includes parameters that can be referred to by one or more SPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each VPS RBSP is initially considered not active at the start of the operation of the decoding process. At most one VPS RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular VPS RBSP results in the deactivation of the previously-active VPS RBSP (if any).

When a VPS RBSP (with a particular value of vps_video_parameter_set_id) is not already active and it is referred to by activation of an SPS RBSP (in which sps_video_parameter_set_id is equal to the vps_video_parameter_set_id value) for the base layer, or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which active_video_parameter_set_id is equal to the vps_video_parameter_set_id value), it is activated. This VPS RBSP is called the active VPS RBSP until it is deactivated by the activation of another VPS RBSP. A VPS RBSP, with that particular value of vps_video_parameter_set_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId equal to 0 or provided through external means, and the VPS NAL unit containing the VPS RBSP shall have nuh_layer_id equal to 0. An activated VPS RBSP shall remain active for the entire CVS.

NOTE 3 – Because an IRAP access unit with NoRaslOutputFlag equal to 1 begins a new CVS and an activated VPS RBSP must remain active for the entire CVS, a VPS RBSP can only be activated by an active parameter sets SEI message when the active parameter sets SEI message is part of an IRAP access unit with NoRaslOutputFlag equal to 1.

Any VPS NAL unit containing the value of vps_video_parameter_set_id for the active VPS RBSP for a CVS shall have the same content as that of the active VPS RBSP for the CVS, unless it follows the last access unit of the CVS and precedes the first VCL NAL unit, the first SPS NAL unit and the first SEI NAL unit containing an active parameter sets SEI message (when present) of another CVS.

NOTE 4 – If VPS RBSP, SPS RBSP or PPS RBSP are conveyed within the bitstream, these constraints impose an order constraint on the NAL units that contain the VPS RBSP, SPS RBSP or PPS RBSP, respectively. Otherwise (VPS RBSP, SPS RBSP or PPS RBSP are conveyed by other means not specified in this Specification), they must be available to the decoding process in a timely fashion such that these constraints are obeyed.

All constraints that are expressed on the relationship between the values of the syntax elements and the values of variables derived from those syntax elements in VPSs, SPSs and PPSs and other syntax elements are expressions of constraints that apply only to the active VPS RBSP, the active SPS RBSP for the base layer and the active PPS RBSP for the base layer. If any VPS RBSP, SPS RBSP and PPS RBSP is present that is never activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it was activated by reference in an otherwise conforming bitstream.

NOTE 5 – In the context of this clause, activation of a parameter set RBSP is for the base layer only. Thus, the constraint above on never-activated parameter set RBSPs applies to those parameter set RBSPs with nuh_layer_id equal to 0 only, because parameter set RBSPs with nuh_layer_id greater than 0 are not allowed to be referred to by the base layer.

During operation of the decoding process (see clause 8), the values of parameters of the active VPS RBSP, the active SPS RBSP for the base layer and the active PPS RBSP for the base layer are considered in effect. For interpretation of SEI messages, the values of the active VPS RBSP, the active SPS RBSP for the base layer and the active PPS RBSP for the base layer for the operation of the decoding process for the VCL NAL units of the coded picture with nuh_layer_id equal to 0 in the same access unit are considered in effect unless otherwise specified in the SEI message semantics.

7.4.2.4.3 Order of access units and association to CVSs

A bitstream conforming to this Specification consists of one or more CVSs.

A CVS consists of one or more access units. The order of NAL units and coded pictures and their association to access units is described in clause 7.4.2.4.4.

The first access unit of a CVS is an IRAP access unit with NoRaslOutputFlag equal to 1.

It is a requirement of bitstream conformance that, when present, the next access unit after an access unit that contains an end of sequence NAL unit shall be an IRAP access unit, which may be an IDR access unit, a BLA access unit, or a CRA access unit.

7.4.2.4.4 Order of NAL units and coded pictures and their association to access units

This clause specifies the order of NAL units and coded pictures and their association to access units for CVSs that conform to one or more of the profiles specified in Annex A and that are decoded using the decoding process specified in clauses 2 through 10.

An access unit consists of one coded picture with nuh_layer_id equal to 0, zero or more VCL NAL units with nuh_layer_id greater than 0 and zero or more non-VCL NAL units. The association of VCL NAL units to coded pictures is described in clause 7.4.2.4.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

Let firstBlPicNalUnit be the first VCL NAL unit of a coded picture with nuh_layer_id equal to 0. The first of any of the following NAL units preceding firstBlPicNalUnit and succeeding the last VCL NAL unit preceding firstBlPicNalUnit, if any, specifies the start of a new access unit:

NOTE 1 – The last VCL NAL unit preceding firstBlPicNalUnit in decoding order may have nuh_layer_id greater than 0.

– access unit delimiter NAL unit with nuh_layer_id equal to 0 (when present),
– VPS NAL unit with nuh_layer_id equal to 0 (when present),
– SPS NAL unit with nuh_layer_id equal to 0 (when present),
– PPS NAL unit with nuh_layer_id equal to 0 (when present),
– Prefix SEI NAL unit with nuh_layer_id equal to 0 (when present),
– NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44 with nuh_layer_id equal to 0 (when present),
– NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 with nuh_layer_id equal to 0 (when present).

NOTE 2 – The first NAL unit preceding firstBlPicNalUnit and succeeding the last VCL NAL unit preceding firstBlPicNalUnit, if any, can only be one of the above-listed NAL units.

When there is none of the above NAL units preceding firstBlPicNalUnit and succeeding the last VCL NAL preceding firstBlPicNalUnit, if any, firstBlPicNalUnit starts a new access unit.

The order of the coded pictures and non-VCL NAL units within an access unit shall obey the following constraints:

– When an access unit delimiter NAL unit with nuh_layer_id equal to 0 is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit with nuh_layer_id equal to 0 in any access unit.
– When any VPS NAL units, SPS NAL units, PPS NAL units, prefix SEI NAL units, NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44, or NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 are present, they shall not follow the last VCL NAL unit of the access unit.
– NAL units having nal_unit_type equal to FD_NUT or SUFFIX_SEI_NUT or in the range of RSV_NVCL45..RSV_NVCL47 or UNSPEC56..UNSPEC63 shall not precede the first VCL NAL unit of the access unit.
– When an end of sequence NAL unit with nuh_layer_id equal to 0 is present, it shall be the last NAL unit among all NAL units with nuh_layer_id equal to 0 in the access unit other than an end of bitstream NAL unit (when present).
– When an end of bitstream NAL unit is present, it shall be the last NAL unit in the access unit.

NOTE 3 – Decoders conforming to profiles specified in Annex A do not use NAL units with nuh_layer_id greater than 0, e.g., access unit delimiter NAL units with nuh_layer_id greater than 0, for access unit boundary detection, except for identification of a NAL unit as a VCL or non-VCL NAL unit.

The structure of access units not containing any NAL units with nal_unit_type equal to FD_NUT, VPS_NUT, SPS_NUT, PPS_NUT, RSV_VCL_N10, RSV_VCL_R11, RSV_VCL_N12, RSV_VCL_R13, RSV_VCL_N14 or RSV_VCL_R15, RSV_IRAP_VCL22 or RSV_IRAP_VCL23, or in the range of RSV_VCL24..RSV_VCL31, RSV_NVCL41..RSV_NVCL47 or UNSPEC48..UNSPEC63 is shown in Figure 7-1.

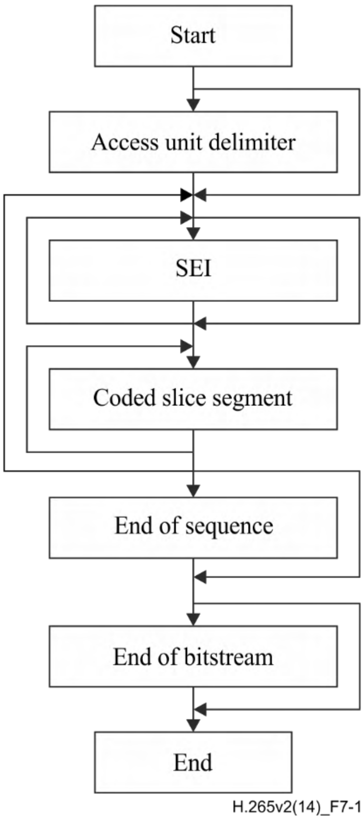


Figure 7-1 – Structure of an access unit not containing any NAL units with nal_unit_type equal to FD_NUT, SUFFIX_SEI_NUT, VPS_NUT, SPS_NUT, PPS_NUT, RSV_VCL_N10, RSV_VCL_R11, RSV_VCL_N12, RSV_VCL_R13, RSV_VCL_N14, RSV_VCL_R15, RSV_IRAP_VCL22 or RSV_IRAP_VCL23, or in the range of RSV_VCL24..RSV_VCL31, RSV_NVCL41..RSV_NVCL47 or UNSPEC48..UNSPEC63

7.4.2.4.5 Order of VCL NAL units and association to coded pictures

This clause specifies the order of VCL NAL units and association to coded pictures.

Each VCL NAL unit is part of a coded picture.

The order of the VCL NAL units within a coded picture is constrained as follows:

– The first VCL NAL unit of the coded picture shall have first_slice_segment_in_pic_flag equal to 1.

– Let sliceSegAddrA and sliceSegAddrB be the slice_segment_address values of any two coded slice segment NAL units A and B within the same coded picture. When either of the following conditions is true, coded slice segment NAL unit A shall precede the coded slice segment NAL unit B:

  – TileId[CtbAddrRsToTs[sliceSegAddrA]] is less than TileId[CtbAddrRsToTs[sliceSegAddrB]].

  – TileId[CtbAddrRsToTs[sliceSegAddrA]] is equal to TileId[CtbAddrRsToTs[sliceSegAddrB]] and CtbAddrRsToTs[sliceSegAddrA] is less than CtbAddrRsToTs[sliceSegAddrB].

7.4.3 Raw byte sequence payloads, trailing bits and byte alignment semantics

7.4.3.1 Video parameter set RBSP semantics

NOTE 1 – VPS NAL units are required to be available to the decoding process prior to their activation (either in the bitstream or by external means), as specified in clause 7.4.2.4.2. However, the VPS RBSP contains information that is not necessary for operation of the decoding process specified in clauses 2 through 10 of this Specification. For purposes other than determining the amount of data in the decoding units of the bitstream (as specified in Annex C), decoders conforming to a profile specified in Annex A but not supporting the INBLD capability specified in Annex F may ignore (remove from the bitstream and discard) the content of all VPS NAL units.

Any two instances of the syntax structure hrd_parameters( ) included in a VPS RBSP shall not have the same content.

vps_video_parameter_set_id identifies the VPS for reference by other syntax elements.

vps_base_layer_internal_flag and **vps_base_layer_available_flag** specify the following:

– If vps_base_layer_internal_flag is equal to 1 and vps_base_layer_available_flag is equal to 1, the base layer is present in the bitstream.
– Otherwise, if vps_base_layer_internal_flag is equal to 0 and vps_base_layer_available_flag is equal to 1, the base layer is provided by an external means not specified in this Specification.
– Otherwise, if vps_base_layer_internal_flag is equal to 1 and vps_base_layer_available_flag is equal to 0, the base layer is not available (neither present in the bitstream nor provided by external means) but the VPS includes information of the base layer as if it were present in the bitstream.
– Otherwise (vps_base_layer_internal_flag is equal to 0 and vps_base_layer_available_flag is equal to 0), the base layer is not available (neither present in the bitstream nor provided by external means) but the VPS includes information of the base layer as if it were provided by an external means not specified in this Specification.

vps_max_layers_minus1 plus 1 specifies the maximum allowed number of layers in each CVS referring to the VPS. It is a requirement of bitstream conformance that, when vps_base_layer_internal_flag is equal to 0, vps_max_layers_minus1 shall be greater than 0. vps_max_layers_minus1 shall be less than 63 in bitstreams conforming to this version of this Specification. The value of 63 for vps_max_layers_minus1 is reserved for future use by ITU-T | ISO/IEC. Although the value of vps_max_layers_minus1 is required to be less than 63 in this version of this Specification, decoders shall allow the value of vps_max_layers_minus1 equal to 63 to appear in the syntax.

NOTE 2 – The value of 63 for vps_max_layers_minus1 may be used to indicate an extended number of layers in a future extension where more than 63 layers in a bitstream need to be supported.

The variable MaxLayersMinus1 is set equal to Min(62, vps_max_layers_minus1).

vps_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the VPS. The value of vps_max_sub_layers_minus1 shall be in the range of 0 to 6, inclusive.

vps_temporal_id_nesting_flag, when vps_max_sub_layers_minus1 is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the VPS. When vps_max_sub_layers_minus1 is equal to 0, vps_temporal_id_nesting_flag shall be equal to 1.

NOTE 3 – The syntax element vps_temporal_id_nesting_flag is used to indicate that temporal sub-layer up-switching, i.e., switching from decoding of up to any TemporalId tIdN to decoding up to any TemporalId tIdM that is greater than tIdN, is always possible.

vps_reserved_0xffff_16bits shall be equal to 0xFFFF in bitstreams conforming to this version of this Specification. Other values for vps_reserved_0xffff_16bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of vps_reserved_0xffff_16bits.

vps_sub_layer_ordering_info_present_flag equal to 1 specifies that vps_max_dec_pic_buffering_minus1[i], vps_max_num_reorder_pics[i] and vps_max_latency_increase_plus1[i] are present for vps_max_sub_layers_minus1 + 1 sub-layers. vps_sub_layer_ordering_info_present_flag equal to 0 specifies that the values of vps_max_dec_pic_buffering_minus1[vps_max_sub_layers_minus1], vps_max_num_reorder_pics[vps_max_sub_layers_minus1] and vps_max_latency_increase_plus1[vps_max_sub_layers_minus1] apply to all sub-layers. When vps_base_layer_internal_flag is equal to 0, vps_sub_layer_ordering_info_present_flag shall be equal to 0 and decoders shall ignore the value of vps_sub_layer_ordering_info_present_flag.

vps_max_dec_pic_buffering_minus1[i] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The value of vps_max_dec_pic_buffering_minus1[i] shall be in the range of 0 to MaxDpbSize − 1 (as specified in clause A.4), inclusive. When i is greater than 0, vps_max_dec_pic_buffering_minus1[i] shall be greater than or equal to vps_max_dec_pic_buffering_minus1[i − 1]. When vps_max_dec_pic_buffering_minus1[i] is not present for i in the range of 0 to vps_max_sub_layers_minus1 − 1, inclusive, due to vps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to vps_max_dec_pic_buffering_minus1[vps_max_sub_layers_minus1]. When vps_base_layer_internal_flag is equal to 0, vps_max_dec_pic_buffering_minus1[i] shall be equal to 0 and decoders shall ignore the value of vps_max_dec_pic_buffering_minus1[i].

vps_max_num_reorder_pics[i] indicates the maximum allowed number of pictures with PicOutputFlag equal to 1 that can precede any picture with PicOutputFlag equal to 1 in the CVS in decoding order and follow that picture with PicOutputFlag equal to 1 in output order when HighestTid is equal to i. The value of vps_max_num_reorder_pics[i] shall be in the range of 0 to vps_max_dec_pic_buffering_minus1[i], inclusive. When i is greater than 0, vps_max_num_reorder_pics[i] shall be greater than or equal to vps_max_num_reorder_pics[i − 1]. When vps_max_num_reorder_pics[i] is not present for i in the range of 0 to vps_max_sub_layers_minus1 − 1, inclusive, due to vps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to vps_max_num_reorder_pics[vps_max_sub_layers_minus1]. When vps_base_layer_internal_flag is equal to 0, vps_max_num_reorder_pics[i] shall be equal to 0 and decoders shall ignore the value of vps_max_num_reorder_pics[i].

vps_max_latency_increase_plus1[i] not equal to 0 is used to compute the value of VpsMaxLatencyPictures[i], which specifies the maximum number of pictures with PicOutputFlag equal to 1 that can precede any picture with PicOutputFlag equal to 1 in the CVS in output order and follow that picture with PicOutputFlag equal to 1 in decoding order when HighestTid is equal to i.

When vps_max_latency_increase_plus1[i] is not equal to 0, the value of VpsMaxLatencyPictures[i] is specified as follows:

$$\text{VpsMaxLatencyPictures[i]} = \text{vps\_max\_num\_reorder\_pics[i]} + \text{vps\_max\_latency\_increase\_plus1[i]} - 1 \quad (7\text{-}2)$$

When vps_max_latency_increase_plus1[i] is equal to 0, no corresponding limit is expressed.

The value of vps_max_latency_increase_plus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. When vps_max_latency_increase_plus1[i] is not present for i in the range of 0 to vps_max_sub_layers_minus1 − 1, inclusive, due to vps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to vps_max_latency_increase_plus1[vps_max_sub_layers_minus1].

When vps_base_layer_internal_flag is equal to 0, vps_max_latency_increase_plus1[i] shall be equal to 0 and decoders shall ignore the value of vps_max_latency_increase_plus1[i].

vps_max_layer_id specifies the maximum allowed value of nuh_layer_id of all NAL units in each CVS referring to the VPS. vps_max_layer_id shall be less than 63 in bitstreams conforming to this version of this Specification. The value of 63 for vps_max_layer_id is reserved for future use by ITU-T | ISO/IEC. Although the value of vps_max_layer_id is required to be less than 63 in this version of this Specification, decoders shall allow a value of vps_max_layer_id equal to 63 to appear in the syntax.

vps_num_layer_sets_minus1 plus 1 specifies the number of layer sets that are specified by the VPS. The value of vps_num_layer_sets_minus1 shall be in the range of 0 to 1 023, inclusive.

layer_id_included_flag[i][j] equal to 1 specifies that the value of nuh_layer_id equal to j is included in the layer identifier list LayerSetLayerIdList[i]. layer_id_included_flag[i][j] equal to 0 specifies that the value of nuh_layer_id equal to j is not included in the layer identifier list LayerSetLayerIdList[i].

The value of NumLayersInIdList[0] is set equal to 1 and the value of LayerSetLayerIdList[0][0] is set equal to 0.

For each value of i in the range of 1 to vps_num_layer_sets_minus1, inclusive, the variable NumLayersInIdList[i] and the layer identifier list LayerSetLayerIdList[i] are derived as follows:

```
n = 0
for( m = 0; m  <=  vps_max_layer_id; m++ )
     if( layer_id_included_flag[ i ][ m ] )                                     (7-3)
          LayerSetLayerIdList[ i ][ n++ ] = m
NumLayersInIdList[ i ] = n
```

For each value of i in the range of 1 to vps_num_layer_sets_minus1, inclusive, NumLayersInIdList[i] shall be in the range of 1 to vps_max_layers_minus1 + 1, inclusive.

When NumLayersInIdList[iA] is equal to NumLayersInIdList[iB] for any iA and iB in the range of 0 to vps_num_layer_sets_minus1, inclusive, with iA not equal to iB, the value of LayerSetLayerIdList[iA][n] shall not be equal to LayerSetLayerIdList[iB][n] for at least one value of n in the range of 0 to NumLayersInIdList[iA], inclusive.

A layer set is identified by the associated layer identifier list. The i-th layer set specified by the VPS is associated with the layer identifier list LayerSetLayerIdList[i], for i in the range of 0 to vps_num_layer_sets_minus1, inclusive.

A layer set consists of all operation points that are associated with the same layer identifier list.

Each operation point is identified by the associated layer identifier list, denoted as OpLayerIdList, which consists of the list of nuh_layer_id values of all NAL units included in the operation point, in increasing order of nuh_layer_id values, and a variable OpTid, which is equal to the highest TemporalId of all NAL units included in the operation point. The bitstream subset associated with the operation point identified by OpLayerIdList and OpTid is the output of the sub-bitstream extraction process as specified in clause 10 with the bitstream, the target highest TemporalId equal to OpTid, and the target layer identifier list equal to OpLayerIdList as inputs. The OpLayerIdList and OpTid that identify an operation point are also referred to as the OpLayerIdList and OpTid associated with the operation point, respectively.

vps_timing_info_present_flag equal to 1 specifies that vps_num_units_in_tick, vps_time_scale, vps_poc_proportional_to_timing_flag and vps_num_hrd_parameters are present in the VPS.

vps_timing_info_present_flag equal to 0 specifies that vps_num_units_in_tick, vps_time_scale, vps_poc_proportional_to_timing_flag and vps_num_hrd_parameters are not present in the VPS.

vps_num_units_in_tick is the number of time units of a clock operating at the frequency vps_time_scale Hz that corresponds to one increment (called a clock tick) of a clock tick counter. The value of vps_num_units_in_tick shall be greater than 0. A clock tick, in units of seconds, is equal to the quotient of vps_num_units_in_tick divided by vps_time_scale. For example, when the picture rate of a video signal is 25 Hz, vps_time_scale may be equal to 27 000 000 and vps_num_units_in_tick may be equal to 1 080 000, and consequently a clock tick may be 0.04 seconds.

vps_time_scale is the number of time units that pass in one second. For example, a time coordinate system that measures time using a 27 MHz clock has a vps_time_scale of 27 000 000. The value of vps_time_scale shall be greater than 0.

vps_poc_proportional_to_timing_flag equal to 1 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, is proportional to the output time of the picture relative to the output time of the first picture in the CVS. vps_poc_proportional_to_timing_flag equal to 0 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, may or may not be proportional to the output time of the picture relative to the output time of the first picture in the CVS.

vps_num_ticks_poc_diff_one_minus1 plus 1 specifies the number of clock ticks corresponding to a difference of picture order count values equal to 1. The value of vps_num_ticks_poc_diff_one_minus1 shall be in the range of 0 to $2^{32} - 2$, inclusive.

vps_num_hrd_parameters specifies the number of hrd_parameters( ) syntax structures present in the VPS RBSP before the vps_extension_flag syntax element. The value of vps_num_hrd_parameters shall be in the range of 0 to vps_num_layer_sets_minus1 + 1, inclusive.

hrd_layer_set_idx[i] specifies the index, into the list of layer sets specified by the VPS, of the layer set to which the i-th hrd_parameters( ) syntax structure in the VPS applies. The value of hrd_layer_set_idx[i] shall be in the range of (vps_base_layer_internal_flag ? 0 : 1) to vps_num_layer_sets_minus1, inclusive.

It is a requirement of bitstream conformance that the value of hrd_layer_set_idx[i] shall not be equal to the value of hrd_layer_set_idx[j] for any value of j not equal to i.

cprms_present_flag[i] equal to 1 specifies that the HRD parameters that are common for all sub-layers are present in the i-th hrd_parameters( ) syntax structure in the VPS. cprms_present_flag[i] equal to 0 specifies that the HRD parameters that are common for all sub-layers are not present in the i-th hrd_parameters( ) syntax structure in the VPS and are derived to be the same as the (i − 1)-th hrd_parameters( ) syntax structure in the VPS. cprms_present_flag[0] is inferred to be equal to 1.

vps_extension_flag equal to 0 specifies that no vps_extension_data_flag syntax elements are present in the VPS RBSP syntax structure. vps_extension_flag equal to 1 specifies that there are vps_extension_data_flag syntax elements present in the VPS RBSP syntax structure. Decoders conforming to a profile specified in Annex A but not supporting the INBLD capability specified in Annex F shall ignore all data that follow the value 1 for vps_extension_flag in a VPS NAL unit.

vps_extension_data_flag may have any value. Its presence and value do not affect the decoding process of the profiles specified in Annex A. Decoders conforming to a profile specified in Annex A but not supporting the INBLD capability specified in Annex F shall ignore all vps_extension_data_flag syntax elements.

7.4.3.2 Sequence parameter set RBSP semantics

7.4.3.2.1 General sequence parameter set RBSP semantics

sps_video_parameter_set_id specifies the value of the vps_video_parameter_set_id of the active VPS.

sps_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the SPS. The value of sps_max_sub_layers_minus1 shall be in the range of 0 to 6, inclusive. The value of sps_max_sub_layers_minus1 shall be less than or equal to vps_max_sub_layers_minus1.

sps_temporal_id_nesting_flag, when sps_max_sub_layers_minus1 is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the SPS. When vps_temporal_id_nesting_flag is equal to 1, sps_temporal_id_nesting_flag shall be equal to 1. When sps_max_sub_layers_minus1 is equal to 0, sps_temporal_id_nesting_flag shall be equal to 1.

NOTE 1 – The syntax element sps_temporal_id_nesting_flag is used to indicate that temporal up-switching, i.e., switching from decoding up to any TemporalId tIdN to decoding up to any TemporalId tIdM that is greater than tIdN, is always possible in the CVS.

sps_seq_parameter_set_id provides an identifier for the SPS for reference by other syntax elements. The value of sps_seq_parameter_set_id shall be in the range of 0 to 15, inclusive.

chroma_format_idc specifies the chroma sampling relative to the luma sampling as specified in clause 6.2. The value of chroma_format_idc shall be in the range of 0 to 3, inclusive.

separate_colour_plane_flag equal to 1 specifies that the three colour components of the 4:4:4 chroma format are coded separately. separate_colour_plane_flag equal to 0 specifies that the colour components are not coded separately. When separate_colour_plane_flag is not present, it is inferred to be equal to 0. When separate_colour_plane_flag is equal to 1, the coded picture consists of three separate components, each of which consists of coded samples of one colour plane (Y, Cb, or Cr) and uses the monochrome coding syntax. In this case, each colour plane is associated with a specific colour_plane_id value.

NOTE 2 – There is no dependency in decoding processes between the colour planes having different colour_plane_id values. For example, the decoding process of a monochrome picture with one value of colour_plane_id does not use any data from monochrome pictures having different values of colour_plane_id for inter prediction.

Depending on the value of separate_colour_plane_flag, the value of the variable ChromaArrayType is assigned as follows:

– If separate_colour_plane_flag is equal to 0, ChromaArrayType is set equal to chroma_format_idc.

– Otherwise (separate_colour_plane_flag is equal to 1), ChromaArrayType is set equal to 0.

pic_width_in_luma_samples specifies the width of each decoded picture in units of luma samples. pic_width_in_luma_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY.

pic_height_in_luma_samples specifies the height of each decoded picture in units of luma samples. pic_height_in_luma_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY.

conformance_window_flag equal to 1 indicates that the conformance cropping window offset parameters follow next in the SPS. conformance_window_flag equal to 0 indicates that the conformance cropping window offset parameters are not present.

conf_win_left_offset, **conf_win_right_offset**, **conf_win_top_offset** and **conf_win_bottom_offset** specify the samples of the pictures in the CVS that are output from the decoding process, in terms of a rectangular region specified in picture coordinates for output. When conformance_window_flag is equal to 0, the values of conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset are inferred to be equal to 0.

The conformance cropping window contains the luma samples with horizontal picture coordinates from SubWidthC * conf_win_left_offset to pic_width_in_luma_samples − (SubWidthC * conf_win_right_offset + 1) and vertical picture coordinates from SubHeightC * conf_win_top_offset to pic_height_in_luma_samples − (SubHeightC * conf_win_bottom_offset + 1), inclusive.

The value of SubWidthC * (conf_win_left_offset + conf_win_right_offset) shall be less than pic_width_in_luma_samples, and the value of SubHeightC * (conf_win_top_offset + conf_win_bottom_offset) shall be less than pic_height_in_luma_samples.

When ChromaArrayType is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates (x / SubWidthC, y / SubHeightC), where (x, y) are the picture coordinates of the specified luma samples.

NOTE 3 – The conformance cropping window offset parameters are only applied at the output. All internal decoding processes are applied to the uncropped picture size.

bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array BitDepth_Y and the value of the luma quantization parameter range offset QpBdOffset_Y as follows:

$$\text{BitDepth}_Y = 8 + \text{bit\_depth\_luma\_minus8} \quad (7\text{-}4)$$

$$\text{QpBdOffset}_Y = 6 * \text{bit\_depth\_luma\_minus8} \quad (7\text{-}5)$$

bit_depth_luma_minus8 shall be in the range of 0 to 8, inclusive.

bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays BitDepth_C and the value of the chroma quantization parameter range offset QpBdOffset_C as follows:

$$\text{BitDepth}_C = 8 + \text{bit\_depth\_chroma\_minus8} \quad (7\text{-}6)$$

$$\text{QpBdOffset}_C = 6 * \text{bit\_depth\_chroma\_minus8} \quad (7\text{-}7)$$

bit_depth_chroma_minus8 shall be in the range of 0 to 8, inclusive.

log2_max_pic_order_cnt_lsb_minus4 specifies the value of the variable MaxPicOrderCntLsb that is used in the decoding process for picture order count as follows:

$$\text{MaxPicOrderCntLsb} = 2^{(\text{ log2\_max\_pic\_order\_cnt\_lsb\_minus4} + 4)} \quad (7\text{-}8)$$

The value of log2_max_pic_order_cnt_lsb_minus4 shall be in the range of 0 to 12, inclusive.

sps_sub_layer_ordering_info_present_flag equal to 1 specifies that sps_max_dec_pic_buffering_minus1[i], sps_max_num_reorder_pics[i] and sps_max_latency_increase_plus1[i] are present for sps_max_sub_layers_minus1 + 1 sub-layers. sps_sub_layer_ordering_info_present_flag equal to 0 specifies that the values of sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1], sps_max_num_reorder_pics[sps_max_sub_layers_minus1] and sps_max_latency_increase_plus1[sps_max_sub_layers_minus1] apply to all sub-layers.

sps_max_dec_pic_buffering_minus1[i] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The value of sps_max_dec_pic_buffering_minus1[i] shall be in the range of 0 to MaxDpbSize − 1, inclusive, where MaxDpbSize is as specified in clause A.4. When i is greater than 0, sps_max_dec_pic_buffering_minus1[i] shall be greater than or equal to sps_max_dec_pic_buffering_minus1[i − 1]. The value of sps_max_dec_pic_buffering_minus1[i] shall be less than or equal to vps_max_dec_pic_buffering_minus1[i] for each value of i. When sps_max_dec_pic_buffering_minus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1].

sps_max_num_reorder_pics[i] indicates the maximum allowed number of pictures with PicOutputFlag equal to 1 that can precede any picture with PicOutputFlag equal to 1 in the CVS in decoding order and follow that picture with PicOutputFlag equal to 1 in output order when HighestTid is equal to i. The value of sps_max_num_reorder_pics[i] shall be in the range of 0 to sps_max_dec_pic_buffering_minus1[i], inclusive. When i is greater than 0, sps_max_num_reorder_pics[i] shall be greater than or equal to sps_max_num_reorder_pics[i − 1]. The value of sps_max_num_reorder_pics[i] shall be less than or equal to vps_max_num_reorder_pics[i] for each value of i. When sps_max_num_reorder_pics[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_num_reorder_pics[sps_max_sub_layers_minus1].

sps_max_latency_increase_plus1[i] not equal to 0 is used to compute the value of SpsMaxLatencyPictures[i], which specifies the maximum number of pictures with PicOutputFlag equal to 1 that can precede any picture with PicOutputFlag equal to 1 in the CVS in output order and follow that picture with PicOutputFlag equal to 1 in decoding order when HighestTid is equal to i.

When sps_max_latency_increase_plus1[i] is not equal to 0, the value of SpsMaxLatencyPictures[i] is specified as follows:

$$\begin{aligned}\text{SpsMaxLatencyPictures[i]} = {} & \text{sps\_max\_num\_reorder\_pics[i]} + \\ & \text{sps\_max\_latency\_increase\_plus1[i]} - 1\end{aligned} \quad (7\text{-}9)$$

When sps_max_latency_increase_plus1[i] is equal to 0, no corresponding limit is expressed.

The value of sps_max_latency_increase_plus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. When vps_max_latency_increase_plus1[i] is not equal to 0, the value of sps_max_latency_increase_plus1[i] shall not be equal to 0 and shall be less than or equal to vps_max_latency_increase_plus1[i] for each value of i. When sps_max_latency_increase_plus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_latency_increase_plus1[sps_max_sub_layers_minus1].

log2_min_luma_coding_block_size_minus3 plus 3 specifies the minimum luma coding block size.

log2_diff_max_min_luma_coding_block_size specifies the difference between the maximum and minimum luma coding block size.

The variables MinCbLog2SizeY, CtbLog2SizeY, MinCbSizeY, CtbSizeY, PicWidthInMinCbsY, PicWidthInCtbsY, PicHeightInMinCbsY, PicHeightInCtbsY, PicSizeInMinCbsY, PicSizeInCtbsY, PicSizeInSamplesY, PicWidthInSamplesC and PicHeightInSamplesC are derived as follows:

$$\text{MinCbLog2SizeY} = \text{log2\_min\_luma\_coding\_block\_size\_minus3} + 3 \quad (7\text{-}10)$$

$$\text{CtbLog2SizeY} = \text{MinCbLog2SizeY} + \text{log2\_diff\_max\_min\_luma\_coding\_block\_size} \quad (7\text{-}11)$$

$$\text{MinCbSizeY} = 1 \ll \text{MinCbLog2SizeY} \quad (7\text{-}12)$$

$$\text{CtbSizeY} = 1 \ll \text{CtbLog2SizeY} \quad (7\text{-}13)$$

$$\text{PicWidthInMinCbsY} = \text{pic\_width\_in\_luma\_samples} / \text{MinCbSizeY} \quad (7\text{-}14)$$

$$\text{PicWidthInCtbsY} = \text{Ceil}(\ \text{pic\_width\_in\_luma\_samples} \div \text{CtbSizeY}\ ) \quad (7\text{-}15)$$

$$\text{PicHeightInMinCbsY} = \text{pic\_height\_in\_luma\_samples} / \text{MinCbSizeY} \quad (7\text{-}16)$$

$$\text{PicHeightInCtbsY} = \text{Ceil}(\ \text{pic\_height\_in\_luma\_samples} \div \text{CtbSizeY}\ ) \quad (7\text{-}17)$$

$$\text{PicSizeInMinCbsY} = \text{PicWidthInMinCbsY} * \text{PicHeightInMinCbsY} \quad (7\text{-}18)$$

$$\text{PicSizeInCtbsY} = \text{PicWidthInCtbsY} * \text{PicHeightInCtbsY} \quad (7\text{-}19)$$

$$\text{PicSizeInSamplesY} = \text{pic\_width\_in\_luma\_samples} * \text{pic\_height\_in\_luma\_samples} \quad (7\text{-}20)$$

$$\text{PicWidthInSamplesC} = \text{pic\_width\_in\_luma\_samples} / \text{SubWidthC} \quad (7\text{-}21)$$

$$\text{PicHeightInSamplesC} = \text{pic\_height\_in\_luma\_samples} / \text{SubHeightC} \quad (7\text{-}22)$$

The variables CtbWidthC and CtbHeightC, which specify the width and height, respectively, of the array for each chroma CTB, are derived as follows:

– If chroma_format_idc is equal to 0 (monochrome) or separate_colour_plane_flag is equal to 1, CtbWidthC and CtbHeightC are both equal to 0.
– Otherwise, CtbWidthC and CtbHeightC are derived as follows:

$$\text{CtbWidthC} = \text{CtbSizeY} / \text{SubWidthC} \quad (7\text{-}23)$$

$$\text{CtbHeightC} = \text{CtbSizeY} / \text{SubHeightC} \quad (7\text{-}24)$$

log2_min_luma_transform_block_size_minus2 plus 2 specifies the minimum luma transform block size.

The variable MinTbLog2SizeY is set equal to log2_min_luma_transform_block_size_minus2 + 2. The CVS shall not contain data that result in MinTbLog2SizeY greater than or equal to MinCbLog2SizeY.

log2_diff_max_min_luma_transform_block_size specifies the difference between the maximum and minimum luma transform block size.

The variable MaxTbLog2SizeY is set equal to log2_min_luma_transform_block_size_minus2 + 2 + log2_diff_max_min_luma_transform_block_size.

The CVS shall not contain data that result in MaxTbLog2SizeY greater than Min(CtbLog2SizeY, 5).

The array ScanOrder[log2BlockSize][scanIdx][sPos][sComp] specifies the mapping of the scan position sPos, ranging from 0 to (1 << log2BlockSize) * (1 << log2BlockSize) − 1, inclusive, to horizontal and vertical components of the scan-order matrix. The array index scanIdx equal to 0 specifies an up-right diagonal scan order, scanIdx equal to 1 specifies a horizontal scan order, scanIdx equal to 2 specifies a vertical scan order, and scanIdx equal to 3 specifies a traverse scan order. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. The array ScanOrder is derived as follows:

For the variable log2BlockSize ranging from 0 to 5, inclusive, the scanning order array ScanOrder is derived as follows:

– For log2BlockSize ranging from 0 to 3, inclusive, the up-right diagonal scan order array initialization process as specified in clause 6.5.3 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][0].
– For log2BlockSize ranging from 0 to 3, inclusive, the horizontal scan order array initialization process as specified in clause 6.5.4 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][1].
– For log2BlockSize ranging from 0 to 3, inclusive, the vertical scan order array initialization process as specified in clause 6.5.5 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][2].
– For log2BlockSize ranging from 2 to 5, inclusive, the traverse scan order array initialization process as specified in clause 6.5.6 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][3].

max_transform_hierarchy_depth_inter specifies the maximum hierarchy depth for transform units of coding units coded in inter prediction mode. The value of max_transform_hierarchy_depth_inter shall be in the range of 0 to CtbLog2SizeY − MinTbLog2SizeY, inclusive.

max_transform_hierarchy_depth_intra specifies the maximum hierarchy depth for transform units of coding units coded in intra prediction mode. The value of max_transform_hierarchy_depth_intra shall be in the range of 0 to CtbLog2SizeY − MinTbLog2SizeY, inclusive.

scaling_list_enabled_flag equal to 1 specifies that a scaling list is used for the scaling process for transform coefficients. scaling_list_enabled_flag equal to 0 specifies that scaling list is not used for the scaling process for transform coefficients.

sps_scaling_list_data_present_flag equal to 1 specifies that the scaling_list_data( ) syntax structure is present in the SPS. sps_scaling_list_data_present_flag equal to 0 specifies that the scaling_list_data( ) syntax structure is not present in the SPS. When not present, the value of sps_scaling_list_data_present_flag is inferred to be equal to 0.

amp_enabled_flag equal to 1 specifies that asymmetric motion partitions, i.e., PartMode equal to PART_2NxnU, PART_2NxnD, PART_nLx2N or PART_nRx2N, may be used in CTBs. amp_enabled_flag equal to 0 specifies that asymmetric motion partitions cannot be used in CTBs.

sample_adaptive_offset_enabled_flag equal to 1 specifies that the sample adaptive offset process is applied to the reconstructed picture after the deblocking filter process. sample_adaptive_offset_enabled_flag equal to 0 specifies that the sample adaptive offset process is not applied to the reconstructed picture after the deblocking filter process.

pcm_enabled_flag equal to 0 specifies that PCM-related syntax (pcm_sample_bit_depth_luma_minus1, pcm_sample_bit_depth_chroma_minus1, log2_min_pcm_luma_coding_block_size_minus3, log2_diff_max_min_pcm_luma_coding_block_size, pcm_loop_filter_disabled_flag, pcm_flag, pcm_alignment_zero_bit syntax elements and pcm_sample( ) syntax structure) is not present in the CVS.

NOTE 4 – When MinCbLog2SizeY is equal to 6 and pcm_enabled_flag is equal to 1, PCM sample data-related syntax (pcm_flag, pcm_alignment_zero_bit syntax elements and pcm_sample( ) syntax structure) is not present in the CVS, because the maximum size of coding blocks that can convey PCM sample data-related syntax is restricted to be less than or equal to Min(CtbLog2SizeY, 5). Hence, MinCbLog2SizeY equal to 6 with pcm_enabled_flag equal to 1 is not an appropriate setting to convey PCM sample data in the CVS.

pcm_sample_bit_depth_luma_minus1 specifies the number of bits used to represent each of PCM sample values of the luma component as follows:

$$\mathrm{PcmBitDepth_Y} = \mathrm{pcm\_sample\_bit\_depth\_luma\_minus1} + 1 \qquad (7\text{-}25)$$

The value of $\mathrm{PcmBitDepth_Y}$ shall be less than or equal to the value of $\mathrm{BitDepth_Y}$.

pcm_sample_bit_depth_chroma_minus1 specifies the number of bits used to represent each of PCM sample values of the chroma components as follows:

$$\mathrm{PcmBitDepth_C} = \mathrm{pcm\_sample\_bit\_depth\_chroma\_minus1} + 1 \qquad (7\text{-}26)$$

The value of $\mathrm{PcmBitDepth_C}$ shall be less than or equal to the value of $\mathrm{BitDepth_C}$. When ChromaArrayType is equal to 0, pcm_sample_bit_depth_chroma_minus1 is not used in the decoding process and decoders shall ignore its value.

log2_min_pcm_luma_coding_block_size_minus3 plus 3 specifies the minimum size of coding blocks with pcm_flag equal to 1.

The variable Log2MinIpcmCbSizeY is set equal to log2_min_pcm_luma_coding_block_size_minus3 + 3. The value of Log2MinIpcmCbSizeY shall be in the range of Min(MinCbLog2SizeY, 5) to Min(CtbLog2SizeY, 5), inclusive.

log2_diff_max_min_pcm_luma_coding_block_size specifies the difference between the maximum and minimum size of coding blocks with pcm_flag equal to 1.

The variable Log2MaxIpcmCbSizeY is set equal to log2_diff_max_min_pcm_luma_coding_block_size + Log2MinIpcmCbSizeY. The value of Log2MaxIpcmCbSizeY shall be less than or equal to Min(CtbLog2SizeY, 5).

pcm_loop_filter_disabled_flag specifies whether the loop filter process is disabled on reconstructed samples in a coding unit with pcm_flag equal to 1 as follows:

– If pcm_loop_filter_disabled_flag is equal to 1, the deblocking filter and sample adaptive offset filter processes on the reconstructed samples in a coding unit with pcm_flag equal to 1 are disabled.
– Otherwise (pcm_loop_filter_disabled_flag value is equal to 0), the deblocking filter and sample adaptive offset filter processes on the reconstructed samples in a coding unit with pcm_flag equal to 1 are not disabled.

When pcm_loop_filter_disabled_flag is not present, it is inferred to be equal to 0.

num_short_term_ref_pic_sets specifies the number of st_ref_pic_set( ) syntax structures included in the SPS. The value of num_short_term_ref_pic_sets shall be in the range of 0 to 64, inclusive.

NOTE 5 – A decoder should allocate memory for a total number of num_short_term_ref_pic_sets + 1 st_ref_pic_set( ) syntax structures since there may be a st_ref_pic_set( ) syntax structure directly signalled in the slice headers of a current picture. A st_ref_pic_set( ) syntax structure directly signalled in the slice headers of a current picture has an index equal to num_short_term_ref_pic_sets.

long_term_ref_pics_present_flag equal to 0 specifies that no long-term reference picture is used for inter prediction of any coded picture in the CVS. long_term_ref_pics_present_flag equal to 1 specifies that long-term reference pictures may be used for inter prediction of one or more coded pictures in the CVS.

num_long_term_ref_pics_sps specifies the number of candidate long-term reference pictures that are specified in the SPS. The value of num_long_term_ref_pics_sps shall be in the range of 0 to 32, inclusive.

lt_ref_pic_poc_lsb_sps[i] specifies the picture order count modulo MaxPicOrderCntLsb of the i-th candidate long-term reference picture specified in the SPS. The number of bits used to represent lt_ref_pic_poc_lsb_sps[i] is equal to log2_max_pic_order_cnt_lsb_minus4 + 4.

used_by_curr_pic_lt_sps_flag[i] equal to 0 specifies that the i-th candidate long-term reference picture specified in the SPS is not used for reference by a picture that includes in its long-term reference picture set (RPS) the i-th candidate long-term reference picture specified in the SPS.

sps_temporal_mvp_enabled_flag equal to 1 specifies that slice_temporal_mvp_enabled_flag is present in the slice headers of non-IDR pictures in the CVS. sps_temporal_mvp_enabled_flag equal to 0 specifies that slice_temporal_mvp_enabled_flag is not present in slice headers and that temporal motion vector predictors are not used in the CVS.

strong_intra_smoothing_enabled_flag equal to 1 specifies that bi-linear interpolation is conditionally used in the intra prediction filtering process in the CVS as specified in clause 8.4.4.2.3. strong_intra_smoothing_enabled_flag equal to 0 specifies that the bi-linear interpolation is not used in the CVS.

vui_parameters_present_flag equal to 1 specifies that the vui_parameters( ) syntax structure as specified in Annex E is present. vui_parameters_present_flag equal to 0 specifies that the vui_parameters( ) syntax structure as specified in Annex E is not present.

sps_extension_present_flag equal to 1 specifies that the syntax elements sps_range_extension_flag, sps_multilayer_extension_flag, sps_3d_extension_flag, sps_scc_extension_flag, and sps_extension_4bits are present in the SPS RBSP syntax structure. sps_extension_present_flag equal to 0 specifies that these syntax elements are not present.

sps_range_extension_flag equal to 1 specifies that the sps_range_extension( ) syntax structure is present in the SPS RBSP syntax structure. sps_range_extension_flag equal to 0 specifies that this syntax structure is not present. When not present, the value of sps_range_extension_flag is inferred to be equal to 0.

sps_multilayer_extension_flag equal to 1 specifies that the sps_multilayer_extension( ) syntax structure (specified in Annex F) is present in the SPS RBSP syntax structure. sps_multilayer_extension_flag equal to 0 specifies that the sps_multilayer_extension( ) syntax structure is not present. When not present, the value of sps_multilayer_extension_flag is inferred to be equal to 0.

sps_3d_extension_flag equal to 1 specifies that the sps_3d_extension( ) syntax structure (specified in Annex I) is present in the SPS RBSP syntax structure. sps_3d_extension_flag equal to 0 specifies that the sps_3d_extension( ) syntax structure is not present. When not present, the value of sps_3d_extension_flag is inferred to be equal to 0.

sps_scc_extension_flag equal to 1 specifies that the sps_scc_extension( ) syntax structure is present in the SPS RBSP syntax structure. sps_scc_extension_flag equal to 0 specifies that this syntax structure is not present. When not present, the value of sps_scc_extension_flag is inferred to be equal to 0.

sps_extension_4bits equal to 0 specifies that no sps_extension_data_flag syntax elements are present in the SPS RBSP syntax structure. When present, sps_extension_4bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of sps_extension_4bits not equal to 0 are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of sps_extension_4bits to be not equal to 0 and shall ignore all sps_extension_data_flag syntax elements in an SPS NAL unit. When not present, the value of sps_extension_4bits is inferred to be equal to 0.

sps_extension_data_flag may have any value. Its presence and value do not affect the decoding process specified in this version of this Specification. Decoders conforming to this version of this Specification shall ignore all sps_extension_data_flag syntax elements.

7.4.3.2.2 Sequence parameter set range extension semantics

transform_skip_rotation_enabled_flag equal to 1 specifies that a rotation is applied to the residual data block for intra 4x4 blocks coded using a transform skip operation. transform_skip_rotation_enabled_flag equal to 0 specifies that this rotation is not applied. When not present, the value of transform_skip_rotation_enabled_flag is inferred to be equal to 0.

transform_skip_context_enabled_flag equal to 1 specifies that a particular context is used for the parsing of the sig_coeff_flag for transform blocks with a skipped transform. transform_skip_context_enabled_flag equal to 0 specifies that the presence or absence of transform skipping or a transform bypass for transform blocks is not used in the context selection for this flag. When not present, the value of transform_skip_context_enabled_flag is inferred to be equal to 0.

implicit_rdpcm_enabled_flag equal to 1 specifies that the residual modification process for blocks using a transform bypass may be used for intra blocks in the CVS. implicit_rdpcm_enabled_flag equal to 0 specifies that the residual modification process is not used for intra blocks in the CVS. When not present, the value of implicit_rdpcm_enabled_flag is inferred to be equal to 0.

explicit_rdpcm_enabled_flag equal to 1 specifies that the residual modification process for blocks using a transform bypass may be used for inter blocks in the CVS. explicit_rdpcm_enabled_flag equal to 0 specifies that the residual modification process is not used for inter blocks in the CVS. When not present, the value of explicit_rdpcm_enabled_flag is inferred to be equal to 0.

extended_precision_processing_flag equal to 1 specifies that an extended dynamic range is used for transform coefficients and transform processing. extended_precision_processing_flag equal to 0 specifies that the extended dynamic range is not used. When not present, the value of extended_precision_processing_flag is inferred to be equal to 0.

The variables CoeffMin_Y, CoeffMin_C, CoeffMax_Y and CoeffMax_C are derived as follows:

$$\text{CoeffMin}_Y = -(1 \ll (\text{extended\_precision\_processing\_flag} \mathbin{?} \text{Max}(15, \text{BitDepth}_Y + 6) : 15)) \quad (7\text{-}27)$$

$$\text{CoeffMin}_C = -(1 \ll (\text{extended\_precision\_processing\_flag} \mathbin{?} \text{Max}(15, \text{BitDepth}_C + 6) : 15)) \quad (7\text{-}28)$$

$$\text{CoeffMax}_Y = (1 \ll (\text{extended\_precision\_processing\_flag} \mathbin{?} \text{Max}(15, \text{BitDepth}_Y + 6) : 15)) - 1 \quad (7\text{-}29)$$

$$\text{CoeffMax}_C = (1 \ll (\text{extended\_precision\_processing\_flag} \mathbin{?} \text{Max}(15, \text{BitDepth}_C + 6) : 15)) - 1 \quad (7\text{-}30)$$

intra_smoothing_disabled_flag equal to 1 specifies that the filtering process of neighbouring samples is unconditionally disabled for intra prediction. intra_smoothing_disabled_flag equal to 0 specifies that the filtering process of neighbouring samples is not disabled. When not present, the value of intra_smoothing_disabled_flag is inferred to be equal to 0.

high_precision_offsets_enabled_flag equal to 1 specifies that weighted prediction offset values are signalled using a bit-depth-dependent precision. high_precision_offsets_enabled_flag equal to 0 specifies that weighted prediction offset values are signalled with a precision equivalent to eight bit processing.

The variables WpOffsetBdShift_Y, WpOffsetBdShift_C, $\text{WpOffsetHalfRange}_Y$ and $\text{WpOffsetHalfRange}_C$ are derived as follows:

$$\text{WpOffsetBdShift}_Y = \text{high\_precision\_offsets\_enabled\_flag} \mathbin{?} 0 : (\text{BitDepth}_Y - 8) \quad (7\text{-}31)$$

$$\text{WpOffsetBdShift}_C = \text{high\_precision\_offsets\_enabled\_flag} \mathbin{?} 0 : (\text{BitDepth}_C - 8) \quad (7\text{-}32)$$

$$\text{WpOffsetHalfRange}_Y = 1 \ll (\text{high\_precision\_offsets\_enabled\_flag} \mathbin{?} (\text{BitDepth}_Y - 1) : 7) \quad (7\text{-}33)$$

$$\text{WpOffsetHalfRange}_C = 1 \ll (\text{high\_precision\_offsets\_enabled\_flag} \mathbin{?} (\text{BitDepth}_C - 1) : 7) \quad (7\text{-}34)$$

persistent_rice_adaptation_enabled_flag equal to 1 specifies that the Rice parameter derivation for the binarization of coeff_abs_level_remaining[ ] is initialized at the start of each sub-block using mode dependent statistics accumulated from previous sub-blocks. persistent_rice_adaptation_enabled_flag equal to 0 specifies that no previous sub-block state is used in Rice parameter derivation. When not present, the value of persistent_rice_adaptation_enabled_flag is inferred to be equal to 0.

cabac_bypass_alignment_enabled_flag equal to 1 specifies that a context-based adaptive binary arithmetic coding (CABAC) alignment process is used prior to bypass decoding of the syntax elements coeff_sign_flag[ ] and coeff_abs_level_remaining[ ]. cabac_bypass_alignment_enabled_flag equal to 0 specifies that no CABAC alignment process is used prior to bypass decoding. When not present, the value of cabac_bypass_alignment_enabled_flag is inferred to be equal to 0.

7.4.3.2.3 Sequence parameter set screen content coding extension semantics

sps_curr_pic_ref_enabled_flag equal to 1 specifies that a picture in the CVS may be included in a reference picture list of a slice of the picture itself. sps_curr_pic_ref_enabled_flag equal to 0 specifies that a picture in the CVS is never included

in a reference picture list of a slice of the picture itself. When not present, the value of sps_curr_pic_ref_enabled_flag is inferred to be equal to 0.

palette_mode_enabled_flag equal to 1 specifies that the decoding process for palette mode may be used for intra blocks. palette_mode_enabled_flag equal to 0 specifies that the decoding process for palette mode is not applied. When not present, the value of palette_mode_enabled_flag is inferred to be equal to 0.

palette_max_size specifies the maximum allowed palette size. When not present, the value of palette_max_size is inferred to be 0.

delta_palette_max_predictor_size specifies the difference between the maximum allowed palette predictor size and the maximum allowed palette size. When not present, the value of delta_palette_max_predictor_size is inferred to be 0. The variable PaletteMaxPredictorSize is derived as follows:

$$\text{PaletteMaxPredictorSize} = \text{palette\_max\_size} + \text{delta\_palette\_max\_predictor\_size} \qquad (7\text{-}35)$$

It is a requirement of bitstream conformance that, when palette_max_size is equal to 0, the value of delta_palette_max_predictor_size shall be equal to 0.

sps_palette_predictor_initializers_present_flag equal to 1 specifies that the sequence palette predictors are initialized using the sps_palette_predictor_initializers. sps_palette_predictor_initializers_present_flag equal to 0 specifies that the entries in the sequence palette predictor are initialized to 0. When not present, the value of sps_palette_predictor_initializers_present_flag is inferred to be equal to 0.

It is a requirement of bitstream conformance that, when palette_max_size is equal to 0, the value of sps_palette_predictor_initializers_present_flag shall be equal to 0.

sps_num_palette_predictor_initializers_minus1 plus 1 specifies the number of entries in the sequence palette predictor initializer.

It is a requirement of bitstream conformance that the value of sps_num_palette_predictor_initializers_minus1 plus 1 shall be less than or equal to PaletteMaxPredictorSize.

sps_palette_predictor_initializer[comp][i] specifies the value of the comp-th component of the i-th palette entry in the SPS that is used to initialize the array PredictorPaletteEntries. For values of i in the range of 0 to sps_num_palette_predictor_initializers_minus1, inclusive, the value of the sps_palette_predictor_initializer[0][i] shall be in the range of 0 to $(1 \ll \text{BitDepth}_Y) - 1$, inclusive, and the values of sps_palette_predictor_initializer[1][i] and sps_palette_predictor_initializer[2][i] shall be in the range of 0 to $(1 \ll \text{BitDepth}_C) - 1$, inclusive.

motion_vector_resolution_control_idc controls the presence and inference of the use_integer_mv_flag that specifies the resolution of motion vectors for inter prediction. The value of motion_vector_resolution_control_idc shall not be equal to 3 in bitstreams conforming to this version of this Specification. The value of 3 for motion_vector_resolution_control_idc is reserved for future use by ITU-T | ISO/IEC. When not present, the value of motion_vector_resolution_control_idc is inferred to be equal to 0.

intra_boundary_filtering_disabled_flag equal to 1 specifies that the intra boundary filtering process is unconditionally disabled for intra prediction. intra_boundary_filtering_disabled_flag equal to 0 specifies that the intra boundary filtering process may be used. When not present, the value of intra_boundary_filtering_disabled_flag is inferred to be equal to 0.

7.4.3.3 Picture parameter set RBSP semantics

7.4.3.3.1 General picture parameter set RBSP semantics

pps_pic_parameter_set_id identifies the PPS for reference by other syntax elements. The value of pps_pic_parameter_set_id shall be in the range of 0 to 63, inclusive.

pps_seq_parameter_set_id specifies the value of sps_seq_parameter_set_id for the active SPS. The value of pps_seq_parameter_set_id shall be in the range of 0 to 15, inclusive.

dependent_slice_segments_enabled_flag equal to 1 specifies the presence of the syntax element dependent_slice_segment_flag in the slice segment headers for coded pictures referring to the PPS. dependent_slice_segments_enabled_flag equal to 0 specifies the absence of the syntax element dependent_slice_segment_flag in the slice segment headers for coded pictures referring to the PPS.

output_flag_present_flag equal to 1 indicates that the pic_output_flag syntax element is present in the associated slice headers. output_flag_present_flag equal to 0 indicates that the pic_output_flag syntax element is not present in the associated slice headers.

num_extra_slice_header_bits specifies the number of extra slice header bits that are present in the slice header RBSP for coded pictures referring to the PPS. The value of num_extra_slice_header_bits shall be in the range of 0 to 2, inclusive, in

bitstreams conforming to this version of this Specification. Other values for num_extra_slice_header_bits are reserved for future use by ITU-T | ISO/IEC. However, decoders shall allow num_extra_slice_header_bits to have any value.

sign_data_hiding_enabled_flag equal to 0 specifies that sign bit hiding is disabled. sign_data_hiding_enabled_flag equal to 1 specifies that sign bit hiding is enabled.

cabac_init_present_flag equal to 1 specifies that cabac_init_flag is present in slice headers referring to the PPS. cabac_init_present_flag equal to 0 specifies that cabac_init_flag is not present in slice headers referring to the PPS.

num_ref_idx_l0_default_active_minus1 specifies the inferred value of num_ref_idx_l0_active_minus1 for P and B slices with num_ref_idx_active_override_flag equal to 0. The value of num_ref_idx_l0_default_active_minus1 shall be in the range of 0 to 14, inclusive.

num_ref_idx_l1_default_active_minus1 specifies the inferred value of num_ref_idx_l1_active_minus1 for B slices with num_ref_idx_active_override_flag equal to 0. The value of num_ref_idx_l1_default_active_minus1 shall be in the range of 0 to 14, inclusive.

init_qp_minus26 plus 26 specifies the initial value of $SliceQp_Y$ for each slice referring to the PPS. The initial value of $SliceQp_Y$ is modified at the slice segment layer when a non-zero value of slice_qp_delta is decoded. The value of init_qp_minus26 shall be in the range of $-(26 + QpBdOffset_Y)$ to +25, inclusive.

constrained_intra_pred_flag equal to 0 specifies that intra prediction allows usage of residual data and decoded samples of neighbouring coding blocks coded using either intra or inter prediction modes. constrained_intra_pred_flag equal to 1 specifies constrained intra prediction, in which case intra prediction only uses residual data and decoded samples from neighbouring coding blocks coded using intra prediction modes.

NOTE 1 – Encoders that operate in error-prone environments should be designed with consideration of the potential for error propagation caused by references to other pictures and references to areas within the current picture that use other pictures as references.

transform_skip_enabled_flag equal to 1 specifies that transform_skip_flag may be present in the residual coding syntax. transform_skip_enabled_flag equal to 0 specifies that transform_skip_flag is not present in the residual coding syntax.

cu_qp_delta_enabled_flag equal to 1 specifies that the diff_cu_qp_delta_depth syntax element is present in the PPS and that cu_qp_delta_abs may be present in the transform unit syntax and the palette syntax. cu_qp_delta_enabled_flag equal to 0 specifies that the diff_cu_qp_delta_depth syntax element is not present in the PPS and that cu_qp_delta_abs is not present in the transform unit syntax and the palette syntax.

diff_cu_qp_delta_depth specifies the difference between the luma CTB size and the minimum luma coding block size of coding units that convey cu_qp_delta_abs and cu_qp_delta_sign_flag. The value of diff_cu_qp_delta_depth shall be in the range of 0 to log2_diff_max_min_luma_coding_block_size, inclusive. When not present, the value of diff_cu_qp_delta_depth is inferred to be equal to 0.

The variable Log2MinCuQpDeltaSize is derived as follows:

$$\text{Log2MinCuQpDeltaSize} = \text{CtbLog2SizeY} - \text{diff\_cu\_qp\_delta\_depth} \quad (7\text{-}36)$$

pps_cb_qp_offset and **pps_cr_qp_offset** specify the offsets to the luma quantization parameter Qp'_Y used for deriving Qp'_{Cb} and Qp'_{Cr}, respectively. The values of pps_cb_qp_offset and pps_cr_qp_offset shall be in the range of −12 to +12, inclusive. When ChromaArrayType is equal to 0, pps_cb_qp_offset and pps_cr_qp_offset are not used in the decoding process and decoders shall ignore their value.

pps_slice_chroma_qp_offsets_present_flag equal to 1 indicates that the slice_cb_qp_offset and slice_cr_qp_offset syntax elements are present in the associated slice headers. pps_slice_chroma_qp_offsets_present_flag equal to 0 indicates that these syntax elements are not present in the associated slice headers. When ChromaArrayType is equal to 0, pps_slice_chroma_qp_offsets_present_flag shall be equal to 0.

weighted_pred_flag equal to 0 specifies that weighted prediction is not applied to P slices. weighted_pred_flag equal to 1 specifies that weighted prediction is applied to P slices.

weighted_bipred_flag equal to 0 specifies that the default weighted prediction is applied to B slices. weighted_bipred_flag equal to 1 specifies that weighted prediction is applied to B slices.

transquant_bypass_enabled_flag equal to 1 specifies that cu_transquant_bypass_flag is present. transquant_bypass_enabled_flag equal to 0 specifies that cu_transquant_bypass_flag is not present.

tiles_enabled_flag equal to 1 specifies that there is more than one tile in each picture referring to the PPS. tiles_enabled_flag equal to 0 specifies that there is only one tile in each picture referring to the PPS.

It is a requirement of bitstream conformance that the value of tiles_enabled_flag shall be the same for all PPSs that are activated within a CVS.

entropy_coding_sync_enabled_flag equal to 1 specifies that a specific synchronization process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is invoked before decoding the CTU which includes the first CTB of a row of CTBs in each tile in each picture referring to the PPS, and a specific storage process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is invoked after decoding the CTU which includes the second CTB of a row of CTBs in each tile in each picture referring to the PPS. entropy_coding_sync_enabled_flag equal to 0 specifies that no specific synchronization process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is required to be invoked before decoding the CTU which includes the first CTB of a row of CTBs in each tile in each picture referring to the PPS, and no specific storage process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is required to be invoked after decoding the CTU which includes the second CTB of a row of CTBs in each tile in each picture referring to the PPS.

It is a requirement of bitstream conformance that the value of entropy_coding_sync_enabled_flag shall be the same for all PPSs that are activated within a CVS.

When entropy_coding_sync_enabled_flag is equal to 1 and the first CTB in a slice is not the first CTB of a row of CTBs in a tile, it is a requirement of bitstream conformance that the last CTB in the slice shall belong to the same row of CTBs as the first CTB in the slice.

When entropy_coding_sync_enabled_flag is equal to 1 and the first CTB in a slice segment is not the first CTB of a row of CTBs in a tile, it is a requirement of bitstream conformance that the last CTB in the slice segment shall belong to the same row of CTBs as the first CTB in the slice segment.

num_tile_columns_minus1 plus 1 specifies the number of tile columns partitioning the picture. num_tile_columns_minus1 shall be in the range of 0 to PicWidthInCtbsY − 1, inclusive. When not present, the value of num_tile_columns_minus1 is inferred to be equal to 0.

num_tile_rows_minus1 plus 1 specifies the number of tile rows partitioning the picture. num_tile_rows_minus1 shall be in the range of 0 to PicHeightInCtbsY − 1, inclusive. When not present, the value of num_tile_rows_minus1 is inferred to be equal to 0.

When tiles_enabled_flag is equal to 1, num_tile_columns_minus1 and num_tile_rows_minus1 shall not be both equal to 0.

uniform_spacing_flag equal to 1 specifies that tile column boundaries and likewise tile row boundaries are distributed uniformly across the picture. uniform_spacing_flag equal to 0 specifies that tile column boundaries and likewise tile row boundaries are not distributed uniformly across the picture but signalled explicitly using the syntax elements column_width_minus1[i] and row_height_minus1[i]. When not present, the value of uniform_spacing_flag is inferred to be equal to 1.

column_width_minus1[i] plus 1 specifies the width of the i-th tile column in units of CTBs.

row_height_minus1[i] plus 1 specifies the height of the i-th tile row in units of CTBs.

The following variables are derived by invoking the CTB raster and tile scanning conversion process as specified in clause 6.5.1:

– The list CtbAddrRsToTs[ctbAddrRs] for ctbAddrRs ranging from 0 to PicSizeInCtbsY − 1, inclusive, specifying the conversion from a CTB address in the CTB raster scan of a picture to a CTB address in the tile scan,
– the list CtbAddrTsToRs[ctbAddrTs] for ctbAddrTs ranging from 0 to PicSizeInCtbsY − 1, inclusive, specifying the conversion from a CTB address in the tile scan to a CTB address in the CTB raster scan of a picture,
– the list TileId[ctbAddrTs] for ctbAddrTs ranging from 0 to PicSizeInCtbsY − 1, inclusive, specifying the conversion from a CTB address in tile scan to a tile ID,
– the list ColumnWidthInLumaSamples[i] for i ranging from 0 to num_tile_columns_minus1, inclusive, specifying the width of the i-th tile column in units of luma samples,
– the list RowHeightInLumaSamples[j] for j ranging from 0 to num_tile_rows_minus1, inclusive, specifying the height of the j-th tile row in units of luma samples.

The values of ColumnWidthInLumaSamples[i] for i ranging from 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] for j ranging from 0 to num_tile_rows_minus1, inclusive, shall all be greater than 0.

The array MinTbAddrZs with elements MinTbAddrZs[x][y] for x ranging from 0 to (PicWidthInCtbsY << (CtbLog2SizeY − MinTbLog2SizeY)) − 1, inclusive, and y ranging from 0 to (PicHeightInCtbsY << (CtbLog2SizeY − MinTbLog2SizeY)) − 1, inclusive, specifying the conversion from a location (x, y) in units of minimum transform blocks to a transform block address in z-scan order, is derived by invoking the z-scan order array initialization process as specified in clause 6.5.2.

loop_filter_across_tiles_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across tile boundaries in pictures referring to the PPS. loop_filter_across_tiles_enabled_flag equal to 0 specifies that in-loop filtering

operations are not performed across tile boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter and sample adaptive offset filter operations. When not present, the value of loop_filter_across_tiles_enabled_flag is inferred to be equal to 1.

pps_loop_filter_across_slices_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across left and upper boundaries of slices referring to the PPS. pps_loop_filter_across_slices_enabled_flag equal to 0 specifies that in-loop filtering operations are not performed across left and upper boundaries of slices referring to the PPS. The in-loop filtering operations include the deblocking filter and sample adaptive offset filter operations.

NOTE 2 – Loop filtering across slice boundaries can be enabled while loop filtering across tile boundaries is disabled and vice versa.

deblocking_filter_control_present_flag equal to 1 specifies the presence of deblocking filter control syntax elements in the PPS. deblocking_filter_control_present_flag equal to 0 specifies the absence of deblocking filter control syntax elements in the PPS.

deblocking_filter_override_enabled_flag equal to 1 specifies the presence of deblocking_filter_override_flag in the slice headers for pictures referring to the PPS. deblocking_filter_override_enabled_flag equal to 0 specifies the absence of deblocking_filter_override_flag in the slice headers for pictures referring to the PPS. When not present, the value of deblocking_filter_override_enabled_flag is inferred to be equal to 0.

pps_deblocking_filter_disabled_flag equal to 1 specifies that the operation of deblocking filter is not applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present. pps_deblocking_filter_disabled_flag equal to 0 specifies that the operation of the deblocking filter is applied for slices referring to the PPS in which slice_deblocking_filter_disabled_flag is not present. When not present, the value of pps_deblocking_filter_disabled_flag is inferred to be equal to 0.

pps_beta_offset_div2 and **pps_tc_offset_div2** specify the default deblocking parameter offsets for β and tC (divided by 2) that are applied for slices referring to the PPS, unless the default deblocking parameter offsets are overridden by the deblocking parameter offsets present in the slice headers of the slices referring to the PPS. The values of pps_beta_offset_div2 and pps_tc_offset_div2 shall both be in the range of −6 to 6, inclusive. When not present, the value of pps_beta_offset_div2 and pps_tc_offset_div2 are inferred to be equal to 0.

pps_scaling_list_data_present_flag equal to 1 specifies that the scaling list data used for the pictures referring to the PPS are derived based on the scaling lists specified by the active SPS and the scaling lists specified by the PPS. pps_scaling_list_data_present_flag equal to 0 specifies that the scaling list data used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS. When scaling_list_enabled_flag is equal to 0, the value of pps_scaling_list_data_present_flag shall be equal to 0. When scaling_list_enabled_flag is equal to 1, sps_scaling_list_data_present_flag is equal to 0 and pps_scaling_list_data_present_flag is equal to 0, the default scaling list data are used to derive the array ScalingFactor as described in the scaling list data semantics as specified in clause 7.4.5.

lists_modification_present_flag equal to 1 specifies that the syntax structure ref_pic_lists_modification( ) is present in the slice segment header. lists_modification_present_flag equal to 0 specifies that the syntax structure ref_pic_lists_modification( ) is not present in the slice segment header.

log2_parallel_merge_level_minus2 plus 2 specifies the value of the variable Log2ParMrgLevel, which is used in the derivation process for luma motion vectors for merge mode as specified in clause 8.5.3.2.2 and the derivation process for spatial merging candidates as specified in clause 8.5.3.2.3. The value of log2_parallel_merge_level_minus2 shall be in the range of 0 to CtbLog2SizeY − 2, inclusive.

The variable Log2ParMrgLevel is derived as follows:

$$\text{Log2ParMrgLevel} = \text{log2\_parallel\_merge\_level\_minus2} + 2 \quad (7\text{-}37)$$

NOTE 3 – The value of Log2ParMrgLevel indicates the built-in capability of parallel derivation of the merging candidate lists. For example, when Log2ParMrgLevel is equal to 6, the merging candidate lists for all the prediction units (PUs) and coding units (CUs) contained in a 64x64 block can be derived in parallel.

slice_segment_header_extension_present_flag equal to 0 specifies that no slice segment header extension syntax elements are present in the slice segment headers for coded pictures referring to the PPS. slice_segment_header_extension_present_flag equal to 1 specifies that slice segment header extension syntax elements are present in the slice segment headers for coded pictures referring to the PPS.

pps_extension_present_flag equal to 1 specifies that the syntax elements pps_range_extension_flag, pps_multilayer_extension_flag, pps_3d_extension_flag, pps_scc_extension_flag, and pps_extension_4bits are present in the picture parameter set RBSP syntax structure. pps_extension_present_flag equal to 0 specifies that these syntax elements are not present.

pps_range_extension_flag equal to 1 specifies that the pps_range_extension( ) syntax structure is present in the PPS RBSP syntax structure. pps_range_extension_flag equal to 0 specifies that this syntax structure is not present. When not present, the value of pps_range_extension_flag is inferred to be equal to 0.

pps_multilayer_extension_flag equal to 1 specifies that the pps_multilayer_extension( ) syntax structure is present in the PPS RBSP syntax structure. pps_multilayer_extension_flag equal to 0 specifies that the pps_multilayer_extension( ) syntax structure is not present. When not present, the value of pps_multilayer_extension_flag is inferred to be equal to 0.

pps_3d_extension_flag equal to 1 specifies that the pps_3d_extension( ) syntax structure (specified in Annex I) is present in the PPS RBSP syntax structure. pps_3d_extension_flag equal to 0 specifies that the pps_3d_extension( ) syntax structure is not present. When not present, the value of pps_3d_extension_flag is inferred to be equal to 0.

pps_scc_extension_flag equal to 1 specifies that the pps_scc_extension( ) syntax structure is present in the PPS RBSP syntax structure. pps_scc_extension_flag equal to 0 specifies that this syntax structure is not present. When not present, the value of pps_scc_extension_flag is inferred to be equal to 0.

pps_extension_4bits equal to 0 specifies that no pps_extension_data_flag syntax elements are present in the PPS RBSP syntax structure. When present, pps_extension_4bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of pps_extension_4bits not equal to 0 are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of pps_extension_4bits to be not equal to 0 and shall ignore all pps_extension_data_flag syntax elements in a PPS NAL unit. When not present, the value of pps_extension_4bits is inferred to be equal to 0.

pps_extension_data_flag may have any value. Its presence and value do not affect the decoding process specified in this version of this Specification. Decoders conforming to this version of this Specification shall ignore all pps_extension_data_flag syntax elements.

7.4.3.3.2 Picture parameter set range extension semantics

log2_max_transform_skip_block_size_minus2 plus 2 specifies the maximum transform block size for which transform_skip_flag may be present in coded pictures referring to the PPS. When not present, the value of log2_max_transform_skip_block_size_minus2 is inferred to be equal to 0. When present, the value of log2_max_transform_skip_block_size_minus2 shall be less than or equal to MaxTbLog2SizeY − 2.

The variable Log2MaxTransformSkipSize is derived as follows:

$$\text{Log2MaxTransformSkipSize} = \text{log2\_max\_transform\_skip\_block\_size\_minus2} + 2 \qquad (7\text{-}38)$$

cross_component_prediction_enabled_flag equal to 1 specifies that log2_res_scale_abs_plus1 and res_scale_sign_flag may be present in the transform unit syntax for pictures referring to the PPS. cross_component_prediction_enabled_flag equal to 0 specifies that log2_res_scale_abs_plus1 and res_scale_sign_flag are not present for pictures referring to the PPS. When not present, the value of cross_component_prediction_enabled_flag is inferred to be equal to 0. When ChromaArrayType is not equal to 3, it is a requirement of bitstream conformance that the value of cross_component_prediction_enabled_flag shall be equal to 0.

chroma_qp_offset_list_enabled_flag equal to 1 specifies that the cu_chroma_qp_offset_flag may be present in the transform unit syntax. chroma_qp_offset_list_enabled_flag equal to 0 specifies that the cu_chroma_qp_offset_flag is not present in the transform unit syntax. When ChromaArrayType is equal to 0, it is a requirement of bitstream conformance that the value of chroma_qp_offset_list_enabled_flag shall be equal to 0.

diff_cu_chroma_qp_offset_depth specifies the difference between the luma CTB size and the minimum luma coding block size of coding units that convey cu_chroma_qp_offset_flag. The value of diff_cu_chroma_qp_offset_depth shall be in the range of 0 to log2_diff_max_min_luma_coding_block_size, inclusive.

The variable Log2MinCuChromaQpOffsetSize is derived as follows:

$$\text{Log2MinCuChromaQpOffsetSize} = \text{CtbLog2SizeY} - \text{diff\_cu\_chroma\_qp\_offset\_depth} \qquad (7\text{-}39)$$

chroma_qp_offset_list_len_minus1 plus 1 specifies the number of cb_qp_offset_list[i] and cr_qp_offset_list[i] syntax elements that are present in the PPS. The value of chroma_qp_offset_list_len_minus1 shall be in the range of 0 to 5, inclusive.

cb_qp_offset_list[i] and **cr_qp_offset_list**[i] specify offsets used in the derivation of Qp'_{Cb} and Qp'_{Cr}, respectively. The values of cb_qp_offset_list[i] and cr_qp_offset_list[i] shall be in the range of −12 to +12, inclusive.

log2_sao_offset_scale_luma is the base 2 logarithm of the scaling parameter that is used to scale sample adaptive offset (SAO) offset values for luma samples. The value of log2_sao_offset_scale_luma shall be in the range of 0 to Max(0, $\text{BitDepth}_{\text{Y}}$ − 10), inclusive. When not present, the value of log2_sao_offset_scale_luma is inferred to be equal to 0.

log2_sao_offset_scale_chroma is the base 2 logarithm of the scaling parameter that is used to scale SAO offset values for chroma samples. The value of log2_sao_offset_scale_chroma shall be in the range of 0 to Max(0, $\text{BitDepth}_C - 10$), inclusive. When not present, the value of log2_sao_offset_scale_chroma is inferred to be equal to 0.

7.4.3.3.3 Picture parameter set screen content coding extension semantics

pps_curr_pic_ref_enabled_flag equal to 1 specifies that a picture referring to the PPS may be included in a reference picture list of a slice of the picture itself. pps_curr_pic_ref_enabled_flag equal to 0 specifies that a picture referring to the PPS is never included in a reference picture list of a slice of the picture itself. When not present, the value of pps_curr_pic_ref_enabled_flag is inferred to be equal to 0.

It is a requirement of bitstream conformance that when sps_curr_pic_ref_enabled_flag is equal to 0, the value of pps_curr_pic_ref_enabled_flag shall be equal to 0.

The variable TwoVersionsOfCurrDecPicFlag is derived as follows:

$$\begin{aligned}&\text{TwoVersionsOfCurrDecPicFlag} = \text{pps\_curr\_pic\_ref\_enabled\_flag}\ \&\& \\ &\quad (\ \text{sample\_adaptive\_offset\_enabled\_flag}\ ||\ !\text{pps\_deblocking\_filter\_disabled\_flag}\ || \\ &\quad \text{deblocking\_filter\_override\_enabled\_flag}\ )\end{aligned} \tag{7-40}$$

When sps_max_dec_pic_buffering_minus1[TemporalId] is equal to 0, the value of TwoVersionsOfCurrDecPicFlag shall be equal to 0.

residual_adaptive_colour_transform_enabled_flag equal to 1 specifies that an adaptive colour transform may be applied to the residual in the decoding process. residual_adaptive_colour_transform_enabled_flag equal to 0 specifies that adaptive colour transform is not applied to the residual. When not present, the value of residual_adaptive_colour_transform_enabled_flag is inferred to be equal to 0.

When ChromaArrayType is not equal to 3, residual_adaptive_colour_transform_enabled_flag shall be equal to 0.

pps_slice_act_qp_offsets_present_flag equal to 1 specifies that slice_act_y_qp_offset, slice_act_cb_qp_offset, slice_act_cr_qp_offset are present in the slice header. pps_slice_act_qp_offsets_present_flag equal to 0 specifies that slice_act_y_qp_offset, slice_act_cb_qp_offset, slice_act_cr_qp_offset are not present in the slice header. When not present, the value of pps_slice_act_qp_offsets_present_flag is inferred to be equal to 0.

pps_act_y_qp_offset_plus5, **pps_act_cb_qp_offset_plus5** and **pps_act_cr_qp_offset_plus3** are used to determine the offsets that are applied to the quantization parameter values qP derived in clause 8.6.2 for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. When not present, the values of pps_act_y_qp_offset_plus5, pps_act_cb_qp_offset_plus5 and pps_act_cr_qp_offset_plus3 are inferred to be equal to 0.

The variable PpsActQpOffsetY is set equal to pps_act_y_qp_offset_plus5 − 5.

The variable PpsActQpOffsetCb is set equal to pps_act_cb_qp_offset_plus5 − 5.

The variable PpsActQpOffsetCr is set equal to pps_act_cb_qp_offset_plus3 − 3.

NOTE – The constant offset values of 5, 5, and 3 above are applied because the adaptive colour transformation that is applied to the residual when tu_residual_act_flag[xTbY][yTbY] is equal to 1 is not an orthonormal transformation.

It is a requirement of bitstream conformance that the values of PpsActQpOffsetY, PpsActQpOffsetCb, and PpsActQpOffsetCr shall be in the range of −12 to +12, inclusive.

pps_palette_predictor_initializers_present_flag equal to 1 specifies that the palette predictor initializers used for the pictures referring to the PPS are derived based on the palette predictor initializers specified by the PPS. pps_palette_predictor_initializer_flag equal to 0 specifies that the palette predictor initializers used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS. When not present, the value of pps_palette_predictor_initializers_present_flag is inferred to be equal to 0.

It is a requirement of bitstream conformance that the value of pps_palette_predictor_initializers_present_flag shall be equal to 0 when either palette_max_size is equal to 0 or palette_mode_enabled_flag is equal to 0.

pps_num_palette_predictor_initializers specifies the number of entries in the picture palette predictor initializer.

It is a requirement of bitstream conformance that the value of pps_num_palette_predictor_initializers shall be less than or equal to PaletteMaxPredictorSize.

monochrome_palette_flag equal to 1 specifies that the pictures that refer to this PPS are monochrome. monochrome_palette_flag equal to 0 specifies that the pictures that refer to this PPS have multiple components.

It is a requirement of bitstream conformance that the value of the monochrome_palette_flag shall be equal to (chroma_format_idc = = 0).

luma_bit_depth_entry_minus8 plus 8 specifies the bit depth of the luma component of the entries of the palette predictor initializer. It is a requirement of bitstream conformance that the value of luma_bit_depth_entry_minus8 shall be equal to the value of bit_depth_luma_minus8.

chroma_bit_depth_entry_minus8 plus 8 specifies the bit depth of the chroma components of the entries of the palette predictor initializer. It is a requirement of bitstream conformance that the value of chroma_bit_depth_entry_minus8 shall be equal to the value of bit_depth_chroma_minus8.

pps_palette_predictor_initializer[comp][i] specifies the value of the comp-th component of the i-th palette entry in the PPS that is used to initialize the array PredictorPaletteEntries. For values of i in the range of 0 to pps_num_palette_predictor_initializers − 1, inclusive, the number of bits used to represent pps_palette_predictor_initializer[0][i] is luma_bit_depth_entry_minus8 + 8, and the number of bits used to represent pps_palette_predictor_initializer[1][i] and pps_palette_predictor_initializer[2][i] is chroma_bit_depth_entry_minus8 + 8.

7.4.3.4 Supplemental enhancement information RBSP semantics

Supplemental enhancement information (SEI) contains information that is not necessary to decode the samples of coded pictures from VCL NAL units. An SEI RBSP contains one or more SEI messages.

7.4.3.5 Access unit delimiter RBSP semantics

The access unit delimiter may be used to indicate the type of slices present in the coded pictures in the access unit containing the access unit delimiter NAL unit and to simplify the detection of the boundary between access units. There is no normative decoding process associated with the access unit delimiter.

pic_type indicates that the slice_type values for all slices of the coded pictures in the access unit containing the access unit delimiter NAL unit are members of the set listed in Table 7-2 for the given value of pic_type. The value of pic_type shall be equal to 0, 1 or 2 in bitstreams conforming to this version of this Specification. Other values of pic_type are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this version of this Specification shall ignore reserved values of pic_type.

Table 7-2 – Interpretation of pic_type

pic_type	slice_type values that may be present in the coded picture
0	I
1	P, I
2	B, P, I

7.4.3.6 End of sequence RBSP semantics

When included in a NAL unit with nuh_layer_id equal to 0, the end of sequence RBSP specifies that the current access unit is the last access unit in the coded video sequence in decoding order and the next subsequent access unit in the bitstream in decoding order (if any) is an IRAP access unit with NoRaslOutputFlag equal to 1. The syntax content of the SODB and RBSP for the end of sequence RBSP are empty.

7.4.3.7 End of bitstream RBSP semantics

The end of bitstream RBSP indicates that no additional NAL units are present in the bitstream that are subsequent to the end of bitstream RBSP in decoding order. The syntax content of the SODB and RBSP for the end of bitstream RBSP are empty.

NOTE – When an elementary stream contains more than one bitstream, the last NAL unit of the last access unit of a bitstream must contain an end of bitstream NAL unit and the first access unit of the subsequent bitstream must be an IRAP access unit. This IRAP access unit may be a CRA, BLA or IDR access unit.

7.4.3.8 Filler data RBSP semantics

The filler data RBSP contains bytes whose value shall be equal to 0xFF. No normative decoding process is specified for a filler data RBSP.

ff_byte is a byte equal to 0xFF.

7.4.3.9 Slice segment layer RBSP semantics

The slice segment layer RBSP consists of a slice segment header and slice segment data.

7.4.3.10 RBSP slice segment trailing bits semantics

cabac_zero_word is a byte-aligned sequence of two bytes equal to 0x0000.

Let NumBytesInVclNalUnits be the sum of the values of NumBytesInNalUnit for all VCL NAL units of a coded picture.

Let BinCountsInNalUnits be the number of times that the parsing process function DecodeBin( ), specified in clause 9.3.4.3, is invoked to decode the contents of all VCL NAL units of a coded picture.

Let the variable RawMinCuBits be derived as follows:

$$\text{RawMinCuBits} = \text{MinCbSizeY} * \text{MinCbSizeY} * (\text{BitDepth}_Y + 2 * \text{BitDepth}_C / (\text{SubWidthC} * \text{SubHeightC})) \quad (7\text{-}41)$$

The value of BinCountsInNalUnits shall be less than or equal to (32 ÷ 3) * NumBytesInVclNalUnits + (RawMinCuBits * PicSizeInMinCbsY) ÷ 32.

NOTE – The constraint on the maximum number of bins resulting from decoding the contents of the coded slice segment NAL units can be met by inserting a number of cabac_zero_word syntax elements to increase the value of NumBytesInVclNalUnits. Each cabac_zero_word is represented in a NAL unit by the three-byte sequence 0x000003 (as a result of the constraints on NAL unit contents that result in requiring inclusion of an emulation_prevention_three_byte for each cabac_zero_word).

7.4.3.11 RBSP trailing bits semantics

rbsp_stop_one_bit shall be equal to 1.

rbsp_alignment_zero_bit shall be equal to 0.

7.4.3.12 Byte alignment semantics

alignment_bit_equal_to_one shall be equal to 1.

alignment_bit_equal_to_zero shall be equal to 0.

7.4.4 Profile, tier and level semantics

general_profile_space specifies the context for the interpretation of general_profile_idc and general_profile_compatibility_flag[j] for all values of j in the range of 0 to 31, inclusive. The value of general_profile_space shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for general_profile_space are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the CVS when general_profile_space is not equal to 0.

general_tier_flag specifies the tier context for the interpretation of general_level_idc as specified in Annex A.

general_profile_idc, when general_profile_space is equal to 0, indicates a profile to which the CVS conforms as specified in Annex A. Bitstreams shall not contain values of general_profile_idc other than those specified in Annex A. Other values of general_profile_idc are reserved for future use by ITU-T | ISO/IEC.

general_profile_compatibility_flag[j] equal to 1, when general_profile_space is equal to 0, indicates that the CVS conforms to the profile indicated by general_profile_idc equal to j as specified in Annex A. When general_profile_space is equal to 0, general_profile_compatibility_flag[general_profile_idc] shall be equal to 1. The value of general_profile_compatibility_flag[j] shall be equal to 0 for any value of j that is not specified as an allowed value of general_profile_idc in Annex A.

general_progressive_source_flag and **general_interlaced_source_flag** are interpreted as follows:

– If general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 0, the source scan type of the pictures in the CVS should be interpreted as progressive only.

– Otherwise, if general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 1, the source scan type of the pictures in the CVS should be interpreted as interlaced only.

– Otherwise, if general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 0, the source scan type of the pictures in the CVS should be interpreted as unknown or unspecified.

– Otherwise (general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 1), the source scan type of each picture in the CVS is indicated at the picture level using the syntax element source_scan_type in a picture timing SEI message.

NOTE 1 – Decoders may ignore the values of general_progressive_source_flag and general_interlaced_source_flag for purposes other than determining the value to be inferred for frame_field_info_present_flag when vui_parameters_present_flag is equal to 0, as there are no other decoding process requirements associated with the values of these flags. Moreover, the actual source scan type of the pictures is outside the scope of this Specification and the method by which the encoder selects the values of general_progressive_source_flag and general_interlaced_source_flag is unspecified.

general_non_packed_constraint_flag equal to 1 specifies that there are no frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, equirectangular projection SEI messages, or cubemap projection SEI messages present in the CVS. general_non_packed_constraint_flag equal to 0 indicates that there may or may not be one or more frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, equirectangular projection SEI messages, or cubemap projection SEI messages present in the CVS.

NOTE 2 – Decoders may ignore the value of general_non_packed_constraint_flag, as there are no decoding process requirements associated with the presence or interpretation of frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, equirectangular projection SEI messages, or cubemap projection SEI messages.

general_frame_only_constraint_flag equal to 1 specifies that field_seq_flag is equal to 0. general_frame_only_constraint_flag equal to 0 indicates that field_seq_flag may or may not be equal to 0.

NOTE 3 – Decoders may ignore the value of general_frame_only_constraint_flag, as there are no decoding process requirements associated with the value of field_seq_flag.

NOTE 4 – When general_progressive_source_flag is equal to 1, general_frame_only_constraint_flag may or may not be equal to 1.

general_max_12bit_constraint_flag, general_max_10bit_constraint_flag, general_max_8bit_constraint_flag, general_max_422chroma_constraint_flag, general_max_420chroma_constraint_flag, general_max_monochrome_constraint_flag, general_intra_constraint_flag, general_one_picture_only_constraint_flag general_lower_bit_rate_constraint_flag, and **general_max_14bit_constraint_flag**, when present, have semantics specified in Annex A when the profile indicated by general_profile_idc or general_profile_compatibility_flag[j] is a profile specified in Annex A.

general_reserved_zero_33bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for general_reserved_zero_33bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of general_reserved_zero_33bits.

general_reserved_zero_34bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for general_reserved_zero_34bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of general_reserved_zero_34bits.

general_reserved_zero_7bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for general_reserved_zero_7bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of general_reserved_zero_7bits.

general_reserved_zero_35bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for general_reserved_zero_35bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of general_reserved_zero_35bits.

general_reserved_zero_43bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for general_reserved_zero_43bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of general_reserved_zero_43bits.

general_inbld_flag equal to 1 specifies that the INBLD capability as specified in Annex F is required for decoding of the layer to which the profile_tier_level( ) syntax structure applies. general_inbld_flag equal to 0 specifies that the INBLD capability as specified in Annex F is not required for decoding of the layer to which the profile_tier_level( ) syntax structure applies. When profilePresentFlag is equal to 1, general_profile_idc is not equal to 9 or 11 and is not in the range of 1 to 5, inclusive, general_profile_compatibility_flag[9] is not equal to 1, general_profile_compatibility_flag[11] is not equal to 1, and general_profile_compatibility_flag[j] is not equal to 1 for any value of j in the range of 1 to 5, inclusive, the value of general_inbld_flag is inferred to be equal to 0.

general_reserved_zero_bit, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. The value 1 for general_reserved_zero_1bit is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of general_reserved_zero_bit.

general_level_idc indicates a level to which the CVS conforms as specified in Annex A. Bitstreams shall not contain values of general_level_idc other than those specified in Annex A. Other values of general_level_idc are reserved for future use by ITU-T | ISO/IEC.

NOTE 5 – A greater value of general_level_idc indicates a higher level. The maximum level signalled in the VPS for a CVS may be higher than the level signalled in the SPS for the same CVS.

NOTE 6 – When the coded video sequence conforms to multiple profiles, general_profile_idc should indicate the profile that provides the preferred decoded result or the preferred bitstream identification, as determined by the encoder (in a manner not specified in this Specification).

NOTE 7 – The syntax elements general_reserved_zero_33bits, general_reserved_zero_34bits and general_reserved_zero_43bits may be used in future versions of this Specification to indicate further constraints on the bitstream (e.g., that a particular syntax combination that would otherwise be permitted by the indicated values of general_profile_compatibility_flag[j], is not used).

sub_layer_profile_present_flag[i] equal to 1, specifies that profile information is present in the profile_tier_level( ) syntax structure for the sub-layer representation with TemporalId equal to i. sub_layer_profile_present_flag[i] equal to 0

specifies that profile information is not present in the profile_tier_level( ) syntax structure for the sub-layer representation with TemporalId equal to i. When profilePresentFlag is equal to 0, sub_layer_profile_present_flag[i] shall be equal to 0.

sub_layer_level_present_flag[i] equal to 1 specifies that level information is present in the profile_tier_level( ) syntax structure for the sub-layer representation with TemporalId equal to i. sub_layer_level_present_flag[i] equal to 0 specifies that level information is not present in the profile_tier_level( ) syntax structure for the sub-layer representation with TemporalId equal to i.

reserved_zero_2bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for reserved_zero_2bits[i] are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of reserved_zero_2bits[i].

Each of the following syntax elements:

– **sub_layer_profile_space**[i],
– **sub_layer_tier_flag**[i],
– **sub_layer_profile_idc**[i],
– **sub_layer_profile_compatibility_flag**[i][j],
– **sub_layer_progressive_source_flag**[i],
– **sub_layer_interlaced_source_flag**[i],
– **sub_layer_non_packed_constraint_flag**[i],
– **sub_layer_frame_only_constraint_flag**[i],
– **sub_layer_max_12bit_constraint_flag**[i],
– **sub_layer_max_10bit_constraint_flag**[i],
– **sub_layer_max_8bit_constraint_flag**[i],
– **sub_layer_max_422chroma_constraint_flag**[i],
– **sub_layer_max_420chroma_constraint_flag**[i],
– **sub_layer_max_monochrome_constraint_flag**[i],
– **sub_layer_intra_constraint_flag**[i],
– **sub_layer_one_picture_only_constraint_flag**[i],
– **sub_layer_lower_bit_rate_constraint_flag**[i],
– **sub_layer_max_14bit_constraint_flag**[i],
– **sub_layer_reserved_zero_33bits**[i],
– **sub_layer_reserved_zero_34bits**[i],
– **sub_layer_reserved_zero_7bits**[i],
– **sub_layer_reserved_zero_35bits**[i],
– **sub_layer_reserved_zero_43bits**[i],
– **sub_layer_inbld_flag**[i],
– **sub_layer_reserved_zero_bit**[i], and
– **sub_layer_level_idc**[i],

is referred to as the i-th corresponding sub-layer syntax element of each of the following syntax elements:

– general_profile_space,
– general_tier_flag,
– general_profile_idc,
– general_profile_compatibility_flag[j],
– general_progressive_source_flag,
– general_interlaced_source_flag,
– general_non_packed_constraint_flag,
– general_frame_only_constraint_flag,
– general_max_12bit_constraint_flag,
– general_max_10bit_constraint_flag,

– general_max_8bit_constraint_flag,
– general_max_422chroma_constraint_flag,
– general_max_420chroma_constraint_flag,
– general_max_monochrome_constraint_flag,
– general_intra_constraint_flag,
– general_one_picture_only_constraint_flag,
– general_lower_bit_rate_constraint_flag,
– general_max_14bit_constraint_flag,
– general_reserved_zero_33bits,
– general_reserved_zero_34bits,
– general_reserved_zero_7bits,
– general_reserved_zero_35bits,
– general_reserved_zero_43bits,
– general_inbld_flag,
– general_reserved_zero_bit, and
– general_level_idc,

respectively.

The semantics of a particular syntax element's i-th corresponding sub-layer syntax element, apart from the specification of the inference of the value when not present, is the same as for the particular syntax element, but applies to the sub-layer representation with TemporalId equal to i.

When not present, the value of sub_layer_tier_flag[i] is inferred to be equal to 0.

NOTE 8 – It is possible that sub_layer_tier_flag[i] is not present and sub_layer_level_idc[i] is present. In this case, a default value of sub_layer_tier_flag[i] is needed for interpretation of sub_layer_level_idc[i].

When the profile_tier_level( ) syntax structure is included in an SPS or is the first profile_tier_level( ) syntax structure in a VPS, and any of the syntax elements sub_layer_profile_space[i], sub_layer_profile_idc[i], sub_layer_profile_compatibility_flag[i][j], sub_layer_progressive_source_flag[i], sub_layer_interlaced_source_flag[i], sub_layer_non_packed_constraint_flag[i], sub_layer_frame_only_constraint_flag[i], sub_layer_max_12bit_constraint_flag[i], sub_layer_max_10bit_constraint_flag[i], sub_layer_max_8bit_constraint_flag[i], sub_layer_max_422chroma_constraint_flag[i], sub_layer_max_420chroma_constraint_flag[i], sub_layer_max_monochrome_constraint_flag[i], sub_layer_intra_constraint_flag[i], sub_layer_one_picture_only_constraint_flag[i], sub_layer_lower_bit_rate_constraint_flag[i], sub_layer_max_14bit_constraint_flag, sub_layer_reserved_zero_33bits[i], sub_layer_reserved_zero_34bits[i], sub_layer_reserved_zero_43bits[i], sub_layer_inbld_flag[i], sub_layer_reserved_zero_1bit[i] and sub_layer_level_idc[i] is not present for any value of i in the range of 0 to maxNumSubLayersMinus1 − 1, inclusive, in the profile_tier_level( ) syntax structure, the value of the syntax element is inferred as follows (in decreasing order of i values from maxNumSubLayersMinus1 − 1 to 0):

– If the value of i is equal to maxNumSubLayersMinus1, the value of the syntax element is inferred to be equal to the value of the corresponding syntax element prefixed with "general_" of the same profile_tier_level( ) syntax structure.

NOTE 9 – For example, in this case, if sub_layer_profile_space[i] is not present, the value is inferred to be equal to general_profile_space of the same profile_tier_level( ) syntax structure.

– Otherwise (the value of i is less than maxNumSubLayersMinus1), the value of the syntax element is inferred to be equal to the corresponding syntax element with i being replaced with i + 1 of the same profile_tier_level( ) syntax structure.

NOTE 10 – For example, in this case, if sub_layer_profile_space[i] is not present, the value is inferred to be equal to sub_layer_profile_space[i + 1] of the same profile_tier_level( ) syntax structure.

7.4.5 Scaling list data semantics

scaling_list_pred_mode_flag[sizeId][matrixId] equal to 0 specifies that the values of the scaling list are the same as the values of a reference scaling list. The reference scaling list is specified by scaling_list_pred_matrix_id_delta[sizeId][matrixId]. scaling_list_pred_mode_flag[sizeId][matrixId] equal to 1 specifies that the values of the scaling list are explicitly signalled.

scaling_list_pred_matrix_id_delta[sizeId][matrixId] specifies the reference scaling list used to derive ScalingList[sizeId][matrixId] as follows:

– If scaling_list_pred_matrix_id_delta[sizeId][matrixId] is equal to 0, the scaling list is inferred from the default scaling list ScalingList[sizeId][matrixId][i] as specified in Table 7-5 and Table 7-6 for i = 0..Min(63, (1 << (4 + (sizeId << 1))) − 1).

– Otherwise, the scaling list is inferred from the reference scaling list as follows:

refMatrixId = matrixId −
scaling_list_pred_matrix_id_delta[sizeId][matrixId] * (sizeId = = 3 ? 3 : 1) (7-42)

ScalingList[sizeId][matrixId][i] = ScalingList[sizeId][refMatrixId][i]
with i =0..Min(63, (1 << (4 + (sizeId << 1))) − 1) (7-43)

If sizeId is less than or equal to 2, the value of scaling_list_pred_matrix_id_delta[sizeId][matrixId] shall be in the range of 0 to matrixId, inclusive. Otherwise (sizeId is equal to 3), the value of scaling_list_pred_matrix_id_delta[sizeId][matrixId] shall be in the range of 0 to matrixId / 3, inclusive.

Table 7-3 – Specification of sizeId

Size of quantization matrix	sizeId
4x4	0
8x8	1
16x16	2
32x32	3

Table 7-4 – Specification of matrixId according to sizeId, prediction mode and colour component

sizeId	CuPredMode	cIdx (Colour component)	matrixId
0, 1, 2, 3	MODE_INTRA	0 (Y)	0
0, 1, 2, 3	MODE_INTRA	1 (Cb)	1
0, 1, 2, 3	MODE_INTRA	2 (Cr)	2
0, 1, 2, 3	MODE_INTER	0 (Y)	3
0, 1, 2, 3	MODE_INTER	1 (Cb)	4
0, 1, 2, 3	MODE_INTER	2 (Cr)	5

scaling_list_dc_coef_minus8[sizeId − 2][matrixId] plus 8 specifies the value of the variable ScalingFactor[2][matrixId][0][0] for the scaling list for the 16x16 size when sizeId is equal to 2 and specifies the value of ScalingFactor[3][matrixId][0][0] for the scaling list for the 32x32 size when sizeId is equal to 3. The value of scaling_list_dc_coef_minus8[sizeId − 2][matrixId] shall be in the range of −7 to 247, inclusive.

When scaling_list_pred_mode_flag[sizeId][matrixId] is equal to 0, scaling_list_pred_matrix_id_delta[sizeId][matrixId] is equal to 0 and sizeId is greater than 1, the value of scaling_list_dc_coef_minus8[sizeId − 2][matrixId] is inferred to be equal to 8.

When scaling_list_pred_matrix_id_delta[sizeId][matrixId] is not equal to 0 and sizeId is greater than 1, the value of scaling_list_dc_coef_minus8[sizeId − 2][matrixId] is inferred to be equal to scaling_list_dc_coef_minus8[sizeId − 2][refMatrixId], where the value of refMatrixId is given by Equation 7-42.

scaling_list_delta_coef specifies the difference between the current matrix coefficient ScalingList[sizeId][matrixId][i] and the previous matrix coefficient ScalingList[sizeId][matrixId][i − 1], when scaling_list_pred_mode_flag[sizeId][matrixId] is equal to 1. The value of scaling_list_delta_coef shall be in the range of −128 to 127, inclusive. The value of ScalingList[sizeId][matrixId][i] shall be greater than 0.

Table 7-5 – Specification of default values of ScalingList[0][matrixId][i] with i = 0..15

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ScalingList[0][0..5][i]	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16

Table 7-6 – Specification of default values of ScalingList[1..3][matrixId][i] with i = 0..63

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ScalingList[1..3][0..2][i]	16	16	16	16	16	16	16	16	16	16	17	16	17	16	17	18
ScalingList[1..3][3..5][i]	16	16	16	16	16	16	16	16	16	16	17	17	17	17	17	18
i − 16	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ScalingList[1..3][0..2][i]	17	18	18	17	18	21	19	20	21	20	19	21	24	22	22	24
ScalingList[1..3][3..5][i]	18	18	18	18	18	20	20	20	20	20	20	20	24	24	24	24
i − 32	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ScalingList[1..3][0..2][i]	24	22	22	24	25	25	27	30	27	25	25	29	31	35	35	31
ScalingList[1..3][3..5][i]	24	24	24	24	25	25	25	25	25	25	25	28	28	28	28	28
i − 48	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ScalingList[1..3][0..2][i]	29	36	41	44	41	36	47	54	54	47	65	70	65	88	88	115
ScalingList[1..3][3..5][i]	28	33	33	33	33	33	41	41	41	41	54	54	54	71	71	91

The four-dimensional array ScalingFactor[sizeId][matrixId][x][y], with x, y = 0..(1 << (2 + sizeId)) − 1, specifies the array of scaling factors according to the variables sizeId specified in Table 7-3 and matrixId specified in Table 7-4.

The elements of the quantization matrix of size 4x4, ScalingFactor[0][matrixId][ ][ ], are derived as follows:

$$\text{ScalingFactor[0][matrixId][x][y] = ScalingList[0][matrixId][i]} \quad (7\text{-}44)$$
$$\text{with i = 0..15, matrixId = 0..5, x = ScanOrder[2][0][i][0], and y = ScanOrder[2][0][i][1]}$$

The elements of the quantization matrix of size 8x8, ScalingFactor[1][matrixId][ ][ ], are derived as follows:

$$\text{ScalingFactor[1][matrixId][x][y] = ScalingList[1][matrixId][i]} \quad (7\text{-}45)$$
$$\text{with i = 0..63, matrixId = 0..5, x = ScanOrder[3][0][i][0], and y = ScanOrder[3][0][i][1]}$$

The elements of the quantization matrix of size 16x16, ScalingFactor[2][matrixId][ ][ ], are derived as follows:

$$\text{ScalingFactor[2][matrixId][x * 2 + k][y * 2 + j] = ScalingList[2][matrixId][i]} \quad (7\text{-}46)$$
$$\text{with i = 0..63, j = 0..1, k = 0..1, matrixId = 0..5, x = ScanOrder[3][0][i][0],}$$
$$\text{and y = ScanOrder[3][0][i][1]}$$

$$\text{ScalingFactor[2][matrixId][0][0] = scaling\_list\_dc\_coef\_minus8[0][matrixId] + 8} \quad (7\text{-}47)$$
$$\text{with matrixId = 0..5}$$

The elements of the quantization matrix of size 32x32, ScalingFactor[3][matrixId][ ][ ], are derived as follows:

$$\text{ScalingFactor[3][matrixId][x * 4 + k][y * 4 + j] = ScalingList[3][matrixId][i]} \quad (7\text{-}48)$$
$$\text{with i = 0..63, j = 0..3, k = 0..3, matrixId = 0, 3, x = ScanOrder[3][0][i][0],}$$
$$\text{and y = ScanOrder[3][0][i][1]}$$

$$\text{ScalingFactor[3][matrixId][0][0] = scaling\_list\_dc\_coef\_minus8[1][matrixId] + 8} \quad (7\text{-}49)$$
$$\text{with matrixId = 0, 3}$$

When ChromaArrayType is equal to 3, the elements of the chroma quantization matrix of size 32x32, ScalingFactor[3][matrixId][ ][ ], with matrixId = 1, 2, 4 and 5 are derived as follows:

$$\text{ScalingFactor[3][matrixId][x * 4 + k][y * 4 + j] = ScalingList[2][matrixId][i]} \quad (7\text{-}50)$$
$$\text{with i = 0..63, j = 0..3, k = 0..3, x = ScanOrder[3][0][i][0], and y = ScanOrder[3][0][i][1]}$$

$$\text{ScalingFactor[3][matrixId][0][0] = scaling\_list\_dc\_coef\_minus8[0][matrixId] + 8} \quad (7\text{-}51)$$

7.4.6 Supplemental enhancement information message semantics

Each SEI message consists of the variables specifying the type payloadType and size payloadSize of the SEI message payload. SEI message payloads are specified in Annex D. The derived SEI message payload size payloadSize is specified in bytes and shall be equal to the number of RBSP bytes in the SEI message payload.

NOTE – The NAL unit byte sequence containing the SEI message might include one or more emulation prevention bytes (represented by emulation_prevention_three_byte syntax elements). Since the payload size of an SEI message is specified in RBSP bytes, the quantity of emulation prevention bytes is not included in the size payloadSize of an SEI payload.

ff_byte is a byte equal to 0xFF identifying a need for a longer representation of the syntax structure that it is used within.

last_payload_type_byte is the last byte of the payload type of an SEI message.

last_payload_size_byte is the last byte of the payload size of an SEI message.

7.4.7 Slice segment header semantics

7.4.7.1 General slice segment header semantics

When present, the value of the slice segment header syntax elements slice_pic_parameter_set_id, pic_output_flag, no_output_of_prior_pics_flag, slice_pic_order_cnt_lsb, short_term_ref_pic_set_sps_flag, short_term_ref_pic_set_idx, num_long_term_sps, num_long_term_pics and slice_temporal_mvp_enabled_flag shall be the same in all slice segment headers of a coded picture. When present, the value of the slice segment header syntax elements lt_idx_sps[i], poc_lsb_lt[i], used_by_curr_pic_lt_flag[i], delta_poc_msb_present_flag[i] and delta_poc_msb_cycle_lt[i] shall be the same in all slice segment headers of a coded picture for each possible value of i.

first_slice_segment_in_pic_flag equal to 1 specifies that the slice segment is the first slice segment of the picture in decoding order. first_slice_segment_in_pic_flag equal to 0 specifies that the slice segment is not the first slice segment of the picture in decoding order.

NOTE 1 – This syntax element may be used for detection of the boundary between coded pictures that are consecutive in decoding order. However, when IDR pictures are consecutive in decoding order and have the same NAL unit type, loss of the first slice of an IDR picture can cause a problem with detection of the boundary between the coded pictures. This can occur, e.g., in the transmission of all-intra-coded video in an error-prone environment. This problem can be mitigated by alternately using the two different IDR NAL unit types (IDR_W_RADL and IDR_N_LP) for any two consecutive IDR pictures. The use of the temporal sub-layer zero index SEI message can also be helpful, as that SEI message includes the syntax element irap_pic_id, the value of which is different for IRAP pictures that are consecutive in decoding order. Some system environments have other provisions that can be helpful for picture boundary detection as well, such as the use of presentation timestamps in Rec. ITU-T H.222.0 | ISO/IEC 13818-1 systems, access unit framing in the ISO/IEC 14496-12 ISO base media file format, or the marker bit in IETF RFC 3550 real-time transport protocol headers.

no_output_of_prior_pics_flag affects the output of previously-decoded pictures in the decoded picture buffer after the decoding of an IDR or a BLA picture that is not the first picture in the bitstream as specified in Annex C.

slice_pic_parameter_set_id specifies the value of pps_pic_parameter_set_id for the PPS in use. The value of slice_pic_parameter_set_id shall be in the range of 0 to 63, inclusive.

Let activePPS be the PPS that has pps_pic_parameter_set_id equal to slice_pic_parameter_set_id, and let activeSPS be the SPS that has sps_seq_parameter_set_id equal to pps_seq_parameter_set_id of activePPS. It is a requirement of bitstream conformance that the following constraints apply:

– The value of TemporalId of activePPS shall be less than or equal to the value of TemporalId of the current picture.

– The value of nuh_layer_id of activePPS shall be less than or equal to the value of nuh_layer_id of the current picture.

– The value of nuh_layer_id of activeSPS shall be less than or equal to the value of nuh_layer_id of the current picture.

dependent_slice_segment_flag equal to 1 specifies that the value of each slice segment header syntax element that is not present is inferred to be equal to the value of the corresponding slice segment header syntax element in the slice header. When not present, the value of dependent_slice_segment_flag is inferred to be equal to 0.

The variable SliceAddrRs is derived as follows:

– If dependent_slice_segment_flag is equal to 0, SliceAddrRs is set equal to slice_segment_address.

– Otherwise, SliceAddrRs is set equal to SliceAddrRs of the preceding slice segment containing the CTB for which the CTB address is CtbAddrTsToRs[CtbAddrRsToTs[slice_segment_address] − 1].

slice_segment_address specifies the address of the first CTB in the slice segment, in CTB raster scan of a picture. The length of the slice_segment_address syntax element is Ceil(Log2(PicSizeInCtbsY)) bits. The value of

slice_segment_address shall be in the range of 0 to PicSizeInCtbsY − 1, inclusive, and the value of slice_segment_address shall not be equal to the value of slice_segment_address of any other coded slice segment NAL unit of the same coded picture. When slice_segment_address is not present, it is inferred to be equal to 0.

The variable CtbAddrInRs, specifying a CTB address in CTB raster scan of a picture, is set equal to slice_segment_address. The variable CtbAddrInTs, specifying a CTB address in tile scan, is set equal to CtbAddrRsToTs[CtbAddrInRs]. The variables $\text{CuQpOffset}_{\text{Cb}}$ and $\text{CuQpOffset}_{\text{Cr}}$, specifying values to be used when determining the respective values of the Qp'_{Cb} and Qp'_{Cr} quantization parameters for the coding unit containing cu_chroma_qp_offset_flag, are both set equal to 0.

slice_reserved_flag[i] has semantics and values that are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the presence and value of slice_reserved_flag[i].

slice_type specifies the coding type of the slice according to Table 7-7.

Table 7-7 – Name association to slice_type

slice_type	Name of slice_type
0	B (B slice)
1	P (P slice)
2	I (I slice)

When nal_unit_type has a value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e., the picture is an IRAP picture, nuh_layer_id is equal to 0, and pps_curr_pic_ref_enabled_flag is equal to 0, slice_type shall be equal to 2.

When sps_max_dec_pic_buffering_minus1[TemporalId] is equal to 0, nuh_layer_id is equal to 0, and pps_curr_pic_ref_enabled_flag is equal to 0, slice_type shall be equal to 2.

pic_output_flag affects the decoded picture output and removal processes as specified in Annex C. When pic_output_flag is not present, it is inferred to be equal to 1.

colour_plane_id specifies the colour plane associated with the current slice RBSP when separate_colour_plane_flag is equal to 1. The value of colour_plane_id shall be in the range of 0 to 2, inclusive. colour_plane_id values 0, 1 and 2 correspond to the Y, Cb and Cr planes, respectively.

NOTE 2 – There is no dependency between the decoding processes of pictures having different values of colour_plane_id.

slice_pic_order_cnt_lsb specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the slice_pic_order_cnt_lsb syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits. The value of the slice_pic_order_cnt_lsb shall be in the range of 0 to MaxPicOrderCntLsb − 1, inclusive. When slice_pic_order_cnt_lsb is not present, slice_pic_order_cnt_lsb is inferred to be equal to 0, except as specified in clause 8.3.3.1.

short_term_ref_pic_set_sps_flag equal to 1 specifies that the short-term RPS of the current picture is derived based on one of the st_ref_pic_set( ) syntax structures in the active SPS that is identified by the syntax element short_term_ref_pic_set_idx in the slice header. short_term_ref_pic_set_sps_flag equal to 0 specifies that the short-term RPS of the current picture is derived based on the st_ref_pic_set( ) syntax structure that is directly included in the slice headers of the current picture. When num_short_term_ref_pic_sets is equal to 0, the value of short_term_ref_pic_set_sps_flag shall be equal to 0.

short_term_ref_pic_set_idx specifies the index, into the list of the st_ref_pic_set( ) syntax structures included in the active SPS, of the st_ref_pic_set( ) syntax structure that is used for derivation of the short-term RPS of the current picture. The syntax element short_term_ref_pic_set_idx is represented by Ceil(Log2(num_short_term_ref_pic_sets)) bits. When not present, the value of short_term_ref_pic_set_idx is inferred to be equal to 0. The value of short_term_ref_pic_set_idx shall be in the range of 0 to num_short_term_ref_pic_sets − 1, inclusive.

The variable CurrRpsIdx is derived as follows:

– If short_term_ref_pic_set_sps_flag is equal to 1, CurrRpsIdx is set equal to short_term_ref_pic_set_idx.
– Otherwise, CurrRpsIdx is set equal to num_short_term_ref_pic_sets.

num_long_term_sps specifies the number of entries in the long-term RPS of the current picture that are derived based on the candidate long-term reference pictures specified in the active SPS. The value of num_long_term_sps shall be in the range of 0 to num_long_term_ref_pics_sps, inclusive. When not present, the value of num_long_term_sps is inferred to be equal to 0.

num_long_term_pics specifies the number of entries in the long-term RPS of the current picture that are directly signalled in the slice header. When not present, the value of num_long_term_pics is inferred to be equal to 0.

When nuh_layer_id is equal to 0, the value of num_long_term_pics shall be less than or equal to sps_max_dec_pic_buffering_minus1[TemporalId] − NumNegativePics[CurrRpsIdx] − NumPositivePics[CurrRpsIdx] − num_long_term_sps − TwoVersionsOfCurrDecPicFlag.

lt_idx_sps[i] specifies an index, into the list of candidate long-term reference pictures specified in the active SPS, of the i-th entry in the long-term RPS of the current picture. The number of bits used to represent lt_idx_sps[i] is equal to Ceil(Log2(num_long_term_ref_pics_sps)). When not present, the value of lt_idx_sps[i] is inferred to be equal to 0. The value of lt_idx_sps[i] shall be in the range of 0 to num_long_term_ref_pics_sps − 1, inclusive.

poc_lsb_lt[i] specifies the value of the picture order count modulo MaxPicOrderCntLsb of the i-th entry in the long-term RPS of the current picture. The length of the poc_lsb_lt[i] syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits.

used_by_curr_pic_lt_flag[i] equal to 0 specifies that the i-th entry in the long-term RPS of the current picture is not used for reference by the current picture.

The variables PocLsbLt[i] and UsedByCurrPicLt[i] are derived as follows:

– If i is less than num_long_term_sps, PocLsbLt[i] is set equal to lt_ref_pic_poc_lsb_sps[lt_idx_sps[i]] and UsedByCurrPicLt[i] is set equal to used_by_curr_pic_lt_sps_flag[lt_idx_sps[i]].
– Otherwise, PocLsbLt[i] is set equal to poc_lsb_lt[i] and UsedByCurrPicLt[i] is set equal to used_by_curr_pic_lt_flag[i].

delta_poc_msb_present_flag[i] equal to 1 specifies that delta_poc_msb_cycle_lt[i] is present. delta_poc_msb_present_flag[i] equal to 0 specifies that delta_poc_msb_cycle_lt[i] is not present.

Let prevTid0Pic be the previous picture in decoding order that has TemporalId equal to 0 and is not a RASL, RADL or SLNR picture. Let setOfPrevPocVals be a set consisting of the following:

– the PicOrderCntVal of prevTid0Pic,
– the PicOrderCntVal of each picture in the RPS of prevTid0Pic,
– the PicOrderCntVal of each picture that follows prevTid0Pic in decoding order and precedes the current picture in decoding order.

When there is more than one value in setOfPrevPocVals for which the value modulo MaxPicOrderCntLsb is equal to PocLsbLt[i], delta_poc_msb_present_flag[i] shall be equal to 1.

delta_poc_msb_cycle_lt[i] is used to determine the value of the most significant bits of the picture order count value of the i-th entry in the long-term RPS of the current picture. The value of delta_poc_msb_cycle_lt[i] shall be in the range of 0 to $2^{(32 - \text{log2\_max\_pic\_order\_cnt\_lsb\_minus4} - 4)}$, inclusive. When delta_poc_msb_cycle_lt[i] is not present, it is inferred to be equal to 0.

The variable DeltaPocMsbCycleLt[i] is derived as follows:

```
if( i = = 0 | | i = = num_long_term_sps )
    DeltaPocMsbCycleLt[ i ] = delta_poc_msb_cycle_lt[ i ]
else                                                                        (7-52)
    DeltaPocMsbCycleLt[ i ] = delta_poc_msb_cycle_lt[ i ] + DeltaPocMsbCycleLt[ i − 1 ]
```

slice_temporal_mvp_enabled_flag specifies whether temporal motion vector predictors can be used for inter prediction. If slice_temporal_mvp_enabled_flag is equal to 0, the syntax elements of the current picture shall be constrained such that no temporal motion vector predictor is used in decoding of the current picture. Otherwise (slice_temporal_mvp_enabled_flag is equal to 1), temporal motion vector predictors may be used in decoding of the current picture. When not present, the value of slice_temporal_mvp_enabled_flag is inferred to be equal to 0.

Let currLayerId be equal to nuh_layer_id of the current NAL unit. When both slice_temporal_mvp_enabled_flag and TemporalId are equal to 0, the syntax elements for all coded pictures with nuh_layer_id equal to currLayerId that follow the current picture in decoding order shall be constrained such that no temporal motion vector from any picture with nuh_layer_id equal to currLayerId that precedes the current picture in decoding order is used in decoding of any coded picture that follows the current picture in decoding order.

NOTE 3 – When slice_temporal_mvp_enabled_flag is equal to 0 in an I slice, it has no impact on the normative decoding process of the picture but merely expresses a bitstream constraint.

NOTE 4 – When slice_temporal_mvp_enabled_flag is equal to 0 in a slice with TemporalId equal to 0, decoders may empty "motion vector storage" for all reference pictures with nuh_layer_id equal to currLayerId in the decoded picture buffer.

slice_sao_luma_flag equal to 1 specifies that SAO is enabled for the luma component in the current slice; slice_sao_luma_flag equal to 0 specifies that SAO is disabled for the luma component in the current slice. When slice_sao_luma_flag is not present, it is inferred to be equal to 0.

slice_sao_chroma_flag equal to 1 specifies that SAO is enabled for the chroma component in the current slice; slice_sao_chroma_flag equal to 0 specifies that SAO is disabled for the chroma component in the current slice. When slice_sao_chroma_flag is not present, it is inferred to be equal to 0.

num_ref_idx_active_override_flag equal to 1 specifies that the syntax element num_ref_idx_l0_active_minus1 is present for P and B slices and that the syntax element num_ref_idx_l1_active_minus1 is present for B slices. num_ref_idx_active_override_flag equal to 0 specifies that the syntax elements num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1 are not present.

num_ref_idx_l0_active_minus1 specifies the maximum reference index for reference picture list 0 that may be used to decode the slice. num_ref_idx_l0_active_minus1 shall be in the range of 0 to 14, inclusive. When the current slice is a P or B slice and num_ref_idx_l0_active_minus1 is not present, num_ref_idx_l0_active_minus1 is inferred to be equal to num_ref_idx_l0_default_active_minus1.

num_ref_idx_l1_active_minus1 specifies the maximum reference index for reference picture list 1 that may be used to decode the slice. num_ref_idx_l1_active_minus1 shall be in the range of 0 to 14, inclusive. When num_ref_idx_l1_active_minus1 is not present, num_ref_idx_l1_active_minus1 is inferred to be equal to num_ref_idx_l1_default_active_minus1.

mvd_l1_zero_flag equal to 1 indicates that the mvd_coding(x0, y0, 1) syntax structure is not parsed and MvdL1[x0][y0][compIdx] is set equal to 0 for compIdx = 0..1. mvd_l1_zero_flag equal to 0 indicates that the mvd_coding(x0, y0, 1) syntax structure is parsed.

cabac_init_flag specifies the method for determining the initialization table used in the initialization process for context variables. When cabac_init_flag is not present, it is inferred to be equal to 0.

collocated_from_l0_flag equal to 1 specifies that the collocated picture used for temporal motion vector prediction is derived from reference picture list 0. collocated_from_l0_flag equal to 0 specifies that the collocated picture used for temporal motion vector prediction is derived from reference picture list 1. When collocated_from_l0_flag is not present, it is inferred to be equal to 1.

collocated_ref_idx specifies the reference index of the collocated picture used for temporal motion vector prediction.

When slice_type is equal to P or when slice_type is equal to B and collocated_from_l0_flag is equal to 1, collocated_ref_idx refers to a picture in list 0, and the value of collocated_ref_idx shall be in the range of 0 to num_ref_idx_l0_active_minus1, inclusive.

When slice_type is equal to B and collocated_from_l0_flag is equal to 0, collocated_ref_idx refers to a picture in list 1 and the value of collocated_ref_idx shall be in the range of 0 to num_ref_idx_l1_active_minus1, inclusive.

When not present, the value of collocated_ref_idx is inferred to be equal to 0.

When slice_temporal_mvp_enabled_flag is equal to 1, it is a requirement of bitstream conformance that, for all slices of the current picture that have slice_type not equal to 2 (if any), the picture referred to by collocated_ref_idx shall be the same and shall not be the current picture.

NOTE 5 – This implies that when pps_curr_pic_ref_enabled_flag is equal to 1 and the current picture is the only reference picture in the reference picture list, slice_temporal_mvp_enabled_flag would be constrained to be equal to 0.

five_minus_max_num_merge_cand specifies the maximum number of merging motion vector prediction (MVP) candidates supported in the slice subtracted from 5. The maximum number of merging MVP candidates, MaxNumMergeCand is derived as follows:

$$\text{MaxNumMergeCand} = 5 - \text{five\_minus\_max\_num\_merge\_cand} \quad (7\text{-}53)$$

The value of MaxNumMergeCand shall be in the range of 1 to 5, inclusive.

use_integer_mv_flag equal to 1 specifies that the resolution of motion vectors for inter prediction in the current slice is integer. use_integer_mv_flag equal to 0 specifies that the resolution of motion vectors for inter prediction in the current slice that refer to pictures other than the current picture is fractional with quarter-sample precision in units of luma samples. When not present, the value of use_integer_mv_flag is inferred to be equal to motion_vector_resolution_control_idc.

slice_qp_delta specifies the initial value of Qp_Y to be used for the coding blocks in the slice until modified by the value of CuQpDeltaVal in the coding unit layer. The initial value of the Qp_Y quantization parameter for the slice, $SliceQp_Y$, is derived as follows:

$$\text{SliceQp}_Y = 26 + \text{init\_qp\_minus26} + \text{slice\_qp\_delta} \quad (7\text{-}54)$$

The value of SliceQp_Y shall be in the range of $-\text{QpBdOffset}_Y$ to +51, inclusive.

slice_cb_qp_offset specifies a difference to be added to the value of pps_cb_qp_offset when determining the value of the Qp'_{Cb} quantization parameter. The value of slice_cb_qp_offset shall be in the range of −12 to +12, inclusive. When slice_cb_qp_offset is not present, it is inferred to be equal to 0. The value of pps_cb_qp_offset + slice_cb_qp_offset shall be in the range of −12 to +12, inclusive.

slice_cr_qp_offset specifies a difference to be added to the value of pps_cr_qp_offset when determining the value of the Qp'_{Cr} quantization parameter. The value of slice_cr_qp_offset shall be in the range of −12 to +12, inclusive. When slice_cr_qp_offset is not present, it is inferred to be equal to 0. The value of pps_cr_qp_offset + slice_cr_qp_offset shall be in the range of −12 to +12, inclusive.

slice_act_y_qp_offset, **slice_act_cb_qp_offset** and **slice_act_cr_qp_offset** specify offsets to the quantization parameter values qP derived in clause 8.6.2 for luma, Cb, and Cr components, respectively. The values of slice_act_y_qp_offset, slice_act_cb_qp_offset and slice_act_cr_qp_offset shall be in the range of −12 to +12, inclusive. When not present, the values of slice_act_y_qp_offset, slice_act_cb_qp_offset, and slice_act_cr_qp_offset are inferred to be equal to 0. The value of PpsActQpOffsetY + slice_act_y_qp_offset shall be in the range of −12 to +12, inclusive. The value of PpsActQpOffsetCb + slice_act_cb_qp_offset shall be in the range of −12 to +12, inclusive. The value of PpsActQpOffsetCr + slice_act_cr_qp_offset shall be in the range of −12 to +12, inclusive.

cu_chroma_qp_offset_enabled_flag equal to 1 specifies that the cu_chroma_qp_offset_flag may be present in the transform unit syntax. cu_chroma_qp_offset_enabled_flag equal to 0 specifies that the cu_chroma_qp_offset_flag is not present in the transform unit syntax. When not present, the value of cu_chroma_qp_offset_enabled_flag is inferred to be equal to 0.

deblocking_filter_override_flag equal to 1 specifies that deblocking parameters are present in the slice header. deblocking_filter_override_flag equal to 0 specifies that deblocking parameters are not present in the slice header. When not present, the value of deblocking_filter_override_flag is inferred to be equal to 0.

slice_deblocking_filter_disabled_flag equal to 1 specifies that the operation of the deblocking filter is not applied for the current slice. slice_deblocking_filter_disabled_flag equal to 0 specifies that the operation of the deblocking filter is applied for the current slice. When slice_deblocking_filter_disabled_flag is not present, it is inferred to be equal to pps_deblocking_filter_disabled_flag.

slice_beta_offset_div2 and **slice_tc_offset_div2** specify the deblocking parameter offsets for β and tC (divided by 2) for the current slice. The values of slice_beta_offset_div2 and slice_tc_offset_div2 shall both be in the range of −6 to 6, inclusive. When not present, the values of slice_beta_offset_div2 and slice_tc_offset_div2 are inferred to be equal to pps_beta_offset_div2 and pps_tc_offset_div2, respectively.

slice_loop_filter_across_slices_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across the left and upper boundaries of the current slice. slice_loop_filter_across_slices_enabled_flag equal to 0 specifies that in-loop operations are not performed across left and upper boundaries of the current slice. The in-loop filtering operations include the deblocking filter and sample adaptive offset filter. When slice_loop_filter_across_slices_enabled_flag is not present, it is inferred to be equal to pps_loop_filter_across_slices_enabled_flag.

num_entry_point_offsets specifies the number of entry_point_offset_minus1[i] syntax elements in the slice header. When not present, the value of num_entry_point_offsets is inferred to be equal to 0.

The value of num_entry_point_offsets is constrained as follows:

– If tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1, the value of num_entry_point_offsets shall be in the range of 0 to PicHeightInCtbsY − 1, inclusive.

– Otherwise, if tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 0, the value of num_entry_point_offsets shall be in the range of 0 to (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1, inclusive.

– Otherwise, when tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 1, the value of num_entry_point_offsets shall be in the range of 0 to (num_tile_columns_minus1 + 1) * PicHeightInCtbsY − 1, inclusive.

offset_len_minus1 plus 1 specifies the length, in bits, of the entry_point_offset_minus1[i] syntax elements. The value of offset_len_minus1 shall be in the range of 0 to 31, inclusive.

entry_point_offset_minus1[i] plus 1 specifies the i-th entry point offset in bytes, and is represented by offset_len_minus1 plus 1 bits. The slice segment data that follow the slice segment header consists of num_entry_point_offsets + 1 subsets, with subset index values ranging from 0 to num_entry_point_offsets, inclusive. The

first byte of the slice segment data is considered byte 0. When present, emulation prevention bytes that appear in the slice segment data portion of the coded slice segment NAL unit are counted as part of the slice segment data for purposes of subset identification. Subset 0 consists of bytes 0 to entry_point_offset_minus1[0], inclusive, of the coded slice segment data, subset k, with k in the range of 1 to num_entry_point_offsets − 1, inclusive, consists of bytes firstByte[k] to lastByte[k], inclusive, of the coded slice segment data with firstByte[k] and lastByte[k] defined as:

$$\text{firstByte}[\,k\,] = \sum_{n=1}^{k}(\text{entry\_point\_offset\_minus1}[n-1]+1) \qquad (7\text{-}55)$$

$$\text{lastByte}[\,k\,] = \text{firstByte}[\,k\,] + \text{entry\_point\_offset\_minus1}[\,k\,] \qquad (7\text{-}56)$$

The last subset (with subset index equal to num_entry_point_offsets) consists of the remaining bytes of the coded slice segment data.

When tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 0, each subset shall consist of all coded bits of all CTUs in the slice segment that are within the same tile, and the number of subsets (i.e., the value of num_entry_point_offsets + 1) shall be equal to the number of tiles that contain CTUs that are in the coded slice segment.

NOTE 6 – When tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 0, each slice must include either a subset of the CTUs of one tile (in which case the syntax element entry_point_offset_minus1[i] is not present) or must include all CTUs of an integer number of complete tiles.

When tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1, each subset k with k in the range of 0 to num_entry_point_offsets, inclusive, shall consist of all coded bits of all CTUs in the slice segment that include luma CTBs that are in the same luma CTB row of the picture, and the number of subsets (i.e., the value of num_entry_point_offsets + 1) shall be equal to the number of CTB rows of the picture that contain CTUs that are in the coded slice segment.

NOTE 7 – The last subset (i.e., subset k for k equal to num_entry_point_offsets) may or may not contain all CTUs that include luma CTBs that are in a luma CTB row of the picture.

When tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 1, each subset k with k in the range of 0 to num_entry_point_offsets, inclusive, shall consist of all coded bits of all CTUs in the slice segment that include luma CTBs that are in the same luma CTB row within a tile, and the number of subsets (i.e., the value of num_entry_point_offsets + 1) shall be equal to the number of luma CTB row scans in the tile scan for the CTUs of the coded slice segment.

slice_segment_header_extension_length specifies the length of the slice segment header extension data in bytes, not including the bits used for signalling slice_segment_header_extension_length itself. The value of slice_segment_header_extension_length shall be in the range of 0 to 256, inclusive. When not present, the value of slice_segment_header_extension_length is inferred to be equal to 0.

slice_segment_header_extension_data_byte may have any value. Decoders shall ignore the value of slice_segment_header_extension_data_byte. Its value does not affect the decoding process of the profiles specified in Annex A.

7.4.7.2 Reference picture list modification semantics

ref_pic_list_modification_flag_l0 equal to 1 indicates that reference picture list 0 is specified explicitly by a list of list_entry_l0[i] values. ref_pic_list_modification_flag_l0 equal to 0 indicates that reference picture list 0 is determined implicitly. When ref_pic_list_modification_flag_l0 is not present in the slice header, it is inferred to be equal to 0.

list_entry_l0[i] specifies the index of the reference picture in RefPicListTemp0 to be placed at the current position of reference picture list 0. The length of the list_entry_l0[i] syntax element is Ceil(Log2(NumPicTotalCurr)) bits. The value of list_entry_l0[i] shall be in the range of 0 to NumPicTotalCurr − 1, inclusive. When the syntax element list_entry_l0[i] is not present in the slice header, it is inferred to be equal to 0.

The variable NumPicTotalCurr is derived as follows:

```
NumPicTotalCurr = 0
for( i = 0; i < NumNegativePics[ CurrRpsIdx ]; i++ )
    if( UsedByCurrPicS0[ CurrRpsIdx ][ i ] )
        NumPicTotalCurr++
for( i = 0; i < NumPositivePics[ CurrRpsIdx ]; i++)                    (7-57)
    if( UsedByCurrPicS1[ CurrRpsIdx ][ i ] )
        NumPicTotalCurr++
for( i = 0; i < num_long_term_sps + num_long_term_pics; i++ )
```

```
        if( UsedByCurrPicLt[ i ] )
            NumPicTotalCurr++
    if( pps_curr_pic_ref_enabled_flag )
        NumPicTotalCurr++
```

ref_pic_list_modification_flag_l1 equal to 1 indicates that reference picture list 1 is specified explicitly by a list of list_entry_l1[i] values. ref_pic_list_modification_flag_l1 equal to 0 indicates that reference picture list 1 is determined implicitly. When ref_pic_list_modification_flag_l1 is not present in the slice header, it is inferred to be equal to 0.

list_entry_l1[i] specifies the index of the reference picture in RefPicListTemp1 to be placed at the current position of reference picture list 1. The length of the list_entry_l1[i] syntax element is Ceil(Log2(NumPicTotalCurr)) bits. The value of list_entry_l1[i] shall be in the range of 0 to NumPicTotalCurr − 1, inclusive. When the syntax element list_entry_l1[i] is not present in the slice header, it is inferred to be equal to 0.

7.4.7.3 Weighted prediction parameters semantics

luma_log2_weight_denom is the base 2 logarithm of the denominator for all luma weighting factors. The value of luma_log2_weight_denom shall be in the range of 0 to 7, inclusive.

delta_chroma_log2_weight_denom is the difference of the base 2 logarithm of the denominator for all chroma weighting factors. When delta_chroma_log2_weight_denom is not present, it is inferred to be equal to 0.

The variable ChromaLog2WeightDenom is derived to be equal to luma_log2_weight_denom + delta_chroma_log2_weight_denom and the value shall be in the range of 0 to 7, inclusive.

luma_weight_l0_flag[i] equal to 1 specifies that weighting factors for the luma component of list 0 prediction using RefPicList0[i] are present. luma_weight_l0_flag[i] equal to 0 specifies that these weighting factors are not present. When luma_weight_l0_flag[i] is not present, it is inferred to be equal to 0.

chroma_weight_l0_flag[i] equal to 1 specifies that weighting factors for the chroma prediction values of list 0 prediction using RefPicList0[i] are present. chroma_weight_l0_flag[i] equal to 0 specifies that these weighting factors are not present. When chroma_weight_l0_flag[i] is not present, it is inferred to be equal to 0.

delta_luma_weight_l0[i] is the difference of the weighting factor applied to the luma prediction value for list 0 prediction using RefPicList0[i].

The variable LumaWeightL0[i] is derived to be equal to (1 << luma_log2_weight_denom) + delta_luma_weight_l0[i]. When luma_weight_l0_flag[i] is equal to 1, the value of delta_luma_weight_l0[i] shall be in the range of −128 to 127, inclusive. When luma_weight_l0_flag[i] is equal to 0, LumaWeightL0[i] is inferred to be equal to $2^{\text{luma\_log2\_weight\_denom}}$.

luma_offset_l0[i] is the additive offset applied to the luma prediction value for list 0 prediction using RefPicList0[i]. The value of luma_offset_l0[i] shall be in the range of $-\text{WpOffsetHalfRange}_Y$ to $\text{WpOffsetHalfRange}_Y - 1$, inclusive. When luma_weight_l0_flag[i] is equal to 0, luma_offset_l0[i] is inferred to be equal to 0.

delta_chroma_weight_l0[i][j] is the difference of the weighting factor applied to the chroma prediction values for list 0 prediction using RefPicList0[i] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaWeightL0[i][j] is derived to be equal to (1 << ChromaLog2WeightDenom) + delta_chroma_weight_l0[i][j]. When chroma_weight_l0_flag[i] is equal to 1, the value of delta_chroma_weight_l0[i][j] shall be in the range of −128 to 127, inclusive. When chroma_weight_l0_flag[i] is equal to 0, ChromaWeightL0[i][j] is inferred to be equal to $2^{\text{ChromaLog2WeightDenom}}$.

delta_chroma_offset_l0[i][j] is the difference of the additive offset applied to the chroma prediction values for list 0 prediction using RefPicList0[i] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaOffsetL0[i][j] is derived as follows:

$$\begin{aligned}&\text{ChromaOffsetL0[i][j]} = \text{Clip3}(-\text{WpOffsetHalfRange}_C, \text{WpOffsetHalfRange}_C - 1,\\&\quad (\text{WpOffsetHalfRange}_C + \text{delta\_chroma\_offset\_l0[i][j]} -\\&\quad ((\text{WpOffsetHalfRange}_C * \text{ChromaWeightL0[i][j]}) >> \text{ChromaLog2WeightDenom})))\end{aligned} \tag{7-58}$$

The value of delta_chroma_offset_l0[i][j] shall be in the range of $-4 * \text{WpOffsetHalfRange}_C$ to $4 * \text{WpOffsetHalfRange}_C - 1$, inclusive. When chroma_weight_l0_flag[i] is equal to 0, ChromaOffsetL0[i][j] is inferred to be equal to 0.

luma_weight_l1_flag[i], **chroma_weight_l1_flag**[i], **delta_luma_weight_l1**[i], **luma_offset_l1**[i], **delta_chroma_weight_l1**[i][j] and **delta_chroma_offset_l1**[i][j] have the same semantics as luma_weight_l0_flag[i], chroma_weight_l0_flag[i], delta_luma_weight_l0[i], luma_offset_l0[i],

delta_chroma_weight_l0[i][j] and delta_chroma_offset_l0[i][j], respectively, with l0, L0, list 0 and List0 replaced by l1, L1, list 1 and List1, respectively.

The variable sumWeightL0Flags is derived to be equal to the sum of luma_weight_l0_flag[i] + 2 * chroma_weight_l0_flag[i], for i = 0..num_ref_idx_l0_active_minus1.

When slice_type is equal to B, the variable sumWeightL1Flags is derived to be equal to the sum of luma_weight_l1_flag[i] + 2 * chroma_weight_l1_flag[i], for i = 0..num_ref_idx_l1_active_minus1.

It is a requirement of bitstream conformance that, when slice_type is equal to P, sumWeightL0Flags shall be less than or equal to 24 and when slice_type is equal to B, the sum of sumWeightL0Flags and sumWeightL1Flags shall be less than or equal to 24.

7.4.8 Short-term reference picture set semantics

The st_ref_pic_set(stRpsIdx) syntax structure may be present in an SPS or in a slice header. Depending on whether the syntax structure is included in a slice header or an SPS, the following applies:

– If present in a slice header, the st_ref_pic_set(stRpsIdx) syntax structure specifies the short-term RPS of the current picture (the picture containing the slice), and the following applies:
  – The content of the st_ref_pic_set(stRpsIdx) syntax structure shall be the same in all slice headers of the current picture.
  – The value of stRpsIdx shall be equal to the syntax element num_short_term_ref_pic_sets in the active SPS.
  – The short-term RPS of the current picture is also referred to as the num_short_term_ref_pic_sets-th candidate short-term RPS in the semantics specified in the remainder of this clause.
– Otherwise (present in an SPS), the st_ref_pic_set(stRpsIdx) syntax structure specifies a candidate short-term RPS, and the term "the current picture" in the semantics specified in the remainder of this clause refers to each picture that has short_term_ref_pic_set_idx equal to stRpsIdx in a CVS that has the SPS as the active SPS.

inter_ref_pic_set_prediction_flag equal to 1 specifies that the stRpsIdx-th candidate short-term RPS is predicted from another candidate short-term RPS, which is referred to as the source candidate short-term RPS. When inter_ref_pic_set_prediction_flag is not present, it is inferred to be equal to 0.

delta_idx_minus1 plus 1 specifies the difference between the value of stRpsIdx and the index, into the list of the candidate short-term RPSs specified in the SPS, of the source candidate short-term RPS. The value of delta_idx_minus1 shall be in the range of 0 to stRpsIdx − 1, inclusive. When delta_idx_minus1 is not present, it is inferred to be equal to 0.

The variable RefRpsIdx is derived as follows:

$$\text{RefRpsIdx} = \text{stRpsIdx} - (\text{ delta\_idx\_minus1} + 1) \qquad (7\text{-}59)$$

delta_rps_sign and **abs_delta_rps_minus1** together specify the value of the variable deltaRps as follows:

$$\text{deltaRps} = (1 - 2 * \text{delta\_rps\_sign}) * (\text{abs\_delta\_rps\_minus1} + 1) \qquad (7\text{-}60)$$

The variable deltaRps represents the value to be added to the picture order count difference values of the source candidate short-term RPS to obtain the picture order count difference values of the stRpsIdx-th candidate short-term RPS. The value of abs_delta_rps_minus1 shall be in the range of 0 to $2^{15} - 1$, inclusive.

used_by_curr_pic_flag[j] equal to 0 specifies that the j-th entry in the source candidate short-term RPS is not used for reference by the current picture.

use_delta_flag[j] equal to 1 specifies that the j-th entry in the source candidate short-term RPS is included in the stRpsIdx-th candidate short-term RPS. use_delta_flag[j] equal to 0 specifies that the j-th entry in the source candidate short-term RPS is not included in the stRpsIdx-th candidate short-term RPS. When use_delta_flag[j] is not present, its value is inferred to be equal to 1.

When inter_ref_pic_set_prediction_flag is equal to 1, the variables DeltaPocS0[stRpsIdx][i], UsedByCurrPicS0[stRpsIdx][i], NumNegativePics[stRpsIdx], DeltaPocS1[stRpsIdx][i], UsedByCurrPicS1[stRpsIdx][i] and NumPositivePics[stRpsIdx] are derived as follows:

```
i = 0
for( j = NumPositivePics[ RefRpsIdx ] − 1; j >= 0; j− − ) {
    dPoc = DeltaPocS1[ RefRpsIdx ][ j ] + deltaRps
```

```
        if( dPoc < 0  &&  use_delta_flag[ NumNegativePics[ RefRpsIdx ] + j ] ) {
            DeltaPocS0[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS0[ stRpsIdx ][ i++ ] =
    used_by_curr_pic_flag[ NumNegativePics[ RefRpsIdx ] + j ]
        }
    }
    if( deltaRps < 0  &&  use_delta_flag[ NumDeltaPocs[ RefRpsIdx ] ] ) {                (7-61)
        DeltaPocS0[ stRpsIdx ][ i ] = deltaRps
        UsedByCurrPicS0[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ NumDeltaPocs[ RefRpsIdx ] ]
    }
    for( j = 0; j < NumNegativePics[ RefRpsIdx ]; j++ ) {
        dPoc = DeltaPocS0[ RefRpsIdx ][ j ] + deltaRps
        if( dPoc < 0  &&  use_delta_flag[ j ] ) {
            DeltaPocS0[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS0[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ j ]
        }
    }
    NumNegativePics[ stRpsIdx ] = i

    i = 0
    for( j = NumNegativePics[ RefRpsIdx ] − 1; j  >=  0; j− − ) {
        dPoc = DeltaPocS0[ RefRpsIdx ][ j ] + deltaRps
        if( dPoc > 0  &&  use_delta_flag[ j ] ) {
            DeltaPocS1[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS1[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ j ]
        }
    }
    if( deltaRps > 0  &&  use_delta_flag[ NumDeltaPocs[ RefRpsIdx ] ] ) {                (7-62)
        DeltaPocS1[ stRpsIdx ][ i ] = deltaRps
        UsedByCurrPicS1[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ NumDeltaPocs[ RefRpsIdx ] ]
    }
    for( j = 0; j < NumPositivePics[ RefRpsIdx ]; j++) {
        dPoc = DeltaPocS1[ RefRpsIdx ][ j ] + deltaRps
        if( dPoc > 0  &&  use_delta_flag[ NumNegativePics[ RefRpsIdx ] + j ] ) {
            DeltaPocS1[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS1[ stRpsIdx ][ i++ ] =
    used_by_curr_pic_flag[ NumNegativePics[ RefRpsIdx ] + j ]
        }
    }
    NumPositivePics[ stRpsIdx ] = i
```

num_negative_pics specifies the number of entries in the stRpsIdx-th candidate short-term RPS that have picture order count values less than the picture order count value of the current picture. When nuh_layer_id of the current picture is equal to 0, the value of num_negative_pics shall be in the range of 0 to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1], inclusive.

num_positive_pics specifies the number of entries in the stRpsIdx-th candidate short-term RPS that have picture order count values greater than the picture order count value of the current picture. When nuh_layer_id of the current picture is equal to 0, the value of num_positive_pics shall be in the range of 0 to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] − num_negative_pics, inclusive.

delta_poc_s0_minus1[i] plus 1, when i is equal to 0, specifies the difference between the picture order count values of the current picture and the i-th entry in the stRpsIdx-th candidate short-term RPS that has picture order count value less than that of the current picture, or, when i is greater than 0, specifies the difference between the picture order count values of the (i − 1)-th entry and the i-th entry in the stRpsIdx-th candidate short-term RPS that have picture order count values less than the picture order count value of the current picture. The value of delta_poc_s0_minus1[i] shall be in the range of 0 to $2^{15} - 1$, inclusive.

used_by_curr_pic_s0_flag[i] equal to 0 specifies that the i-th entry in the stRpsIdx-th candidate short-term RPS that has picture order count value less than that of the current picture is not used for reference by the current picture.

delta_poc_s1_minus1[i] plus 1, when i is equal to 0, specifies the difference between the picture order count values of the current picture and the i-th entry in the stRpsIdx-th candidate short-term RPS that has picture order count value greater than that of the current picture, or, when i is greater than 0, specifies the difference between the picture order count values of the i-th entry and the (i − 1)-th entry in the current candidate short-term RPS that have picture order count values greater than the picture order count value of the current picture. The value of delta_poc_s1_minus1[i] shall be in the range of 0 to $2^{15} - 1$, inclusive.

used_by_curr_pic_s1_flag[i] equal to 0 specifies that the i-th entry in the current candidate short-term RPS that has picture order count value greater than that of the current picture is not used for reference by the current picture.

When inter_ref_pic_set_prediction_flag is equal to 0, the variables NumNegativePics[stRpsIdx], NumPositivePics[stRpsIdx], UsedByCurrPicS0[stRpsIdx][i], UsedByCurrPicS1[stRpsIdx][i], DeltaPocS0[stRpsIdx][i] and DeltaPocS1[stRpsIdx][i] are derived as follows:

NumNegativePics[stRpsIdx] = num_negative_pics (7-63)

NumPositivePics[stRpsIdx] = num_positive_pics (7-64)

UsedByCurrPicS0[stRpsIdx][i] = used_by_curr_pic_s0_flag[i] (7-65)

UsedByCurrPicS1[stRpsIdx][i] = used_by_curr_pic_s1_flag[i] (7-66)

– If i is equal to 0, the following applies:

DeltaPocS0[stRpsIdx][i] = −(delta_poc_s0_minus1[i] + 1) (7-67)

DeltaPocS1[stRpsIdx][i] = delta_poc_s1_minus1[i] + 1 (7-68)

– Otherwise, the following applies:

DeltaPocS0[stRpsIdx][i] = DeltaPocS0[stRpsIdx][i − 1] − (delta_poc_s0_minus1[i] + 1) (7-69)

DeltaPocS1[stRpsIdx][i] = DeltaPocS1[stRpsIdx][i − 1] + (delta_poc_s1_minus1[i] + 1) (7-70)

The variable NumDeltaPocs[stRpsIdx] is derived as follows:

NumDeltaPocs[stRpsIdx] = NumNegativePics[stRpsIdx] + NumPositivePics[stRpsIdx] (7-71)

7.4.9 Slice segment data semantics

7.4.9.1 General slice segment data semantics

end_of_slice_segment_flag equal to 0 specifies that another CTU is following in the slice. end_of_slice_segment_flag equal to 1 specifies the end of the slice segment, i.e., that no further CTU follows in the slice segment.

end_of_subset_one_bit shall be equal to 1.

7.4.9.2 Coding tree unit semantics

The CTU is the root node of the coding quadtree structure.

7.4.9.3 Sample adaptive offset semantics

sao_merge_left_flag equal to 1 specifies that the syntax elements sao_type_idx_luma, sao_type_idx_chroma, sao_band_position, sao_eo_class_luma, sao_eo_class_chroma, sao_offset_abs and sao_offset_sign are derived from the corresponding syntax elements of the left CTB. sao_merge_left_flag equal to 0 specifies that these syntax elements are not derived from the corresponding syntax elements of the left CTB. When sao_merge_left_flag is not present, it is inferred to be equal to 0.

sao_merge_up_flag equal to 1 specifies that the syntax elements sao_type_idx_luma, sao_type_idx_chroma, sao_band_position, sao_eo_class_luma, sao_eo_class_chroma, sao_offset_abs and sao_offset_sign are derived from the corresponding syntax elements of the above CTB. sao_merge_up_flag equal to 0 specifies that these syntax elements are

not derived from the corresponding syntax elements of the above CTB. When sao_merge_up_flag is not present, it is inferred to be equal to 0.

sao_type_idx_luma specifies the offset type for the luma component. The array SaoTypeIdx[cIdx][rx][ry] specifies the offset type as specified in Table 7-8 for the CTB at the location (rx, ry) for the colour component cIdx. The value of SaoTypeIdx[0][rx][ry] is derived as follows:

– If sao_type_idx_luma is present, SaoTypeIdx[0][rx][ry] is set equal to sao_type_idx_luma.
– Otherwise (sao_type_idx_luma is not present), SaoTypeIdx[0][rx][ry] is derived as follows:
  – If sao_merge_left_flag is equal to 1, SaoTypeIdx[0][rx][ry] is set equal to SaoTypeIdx[0][rx − 1][ry].
  – Otherwise, if sao_merge_up_flag is equal to 1, SaoTypeIdx[0][rx][ry] is set equal to SaoTypeIdx[0][rx][ry − 1].
  – Otherwise, SaoTypeIdx[0][rx][ry] is set equal to 0.

sao_type_idx_chroma specifies the offset type for the chroma components. The values of SaoTypeIdx[cIdx][rx][ry] are derived as follows for cIdx equal to 1..2:

– If sao_type_idx_chroma is present, SaoTypeIdx[cIdx][rx][ry] is set equal to sao_type_idx_chroma.
– Otherwise (sao_type_idx_chroma is not present), SaoTypeIdx[cIdx][rx][ry] is derived as follows:
  – If sao_merge_left_flag is equal to 1, SaoTypeIdx[cIdx][rx][ry] is set equal to SaoTypeIdx[cIdx][rx − 1][ry].
  – Otherwise, if sao_merge_up_flag is equal to 1, SaoTypeIdx[cIdx][rx][ry] is set equal to SaoTypeIdx[cIdx][rx][ry − 1].
  – Otherwise, SaoTypeIdx[cIdx][rx][ry] is set equal to 0.

Table 7-8 – Specification of the SAO type

SaoTypeIdx[cIdx][rx][ry]	SAO type (informative)
0	Not applied
1	Band offset
2	Edge offset

sao_offset_abs[cIdx][rx][ry][i] specifies the offset value of i-th category for the CTB at the location (rx, ry) for the colour component cIdx.

When sao_offset_abs[cIdx][rx][ry][i] is not present, it is inferred as follows:

– If sao_merge_left_flag is equal to 1, sao_offset_abs[cIdx][rx][ry][i] is inferred to be equal to sao_offset_abs[cIdx][rx − 1][ry][i].
– Otherwise, if sao_merge_up_flag is equal to 1, sao_offset_abs[cIdx][rx][ry][i] is inferred to be equal to sao_offset_abs[cIdx][rx][ry − 1][i].
– Otherwise, sao_offset_abs[cIdx][rx][ry][i] is inferred to be equal to 0.

sao_offset_sign[cIdx][rx][ry][i] specifies the sign of the offset value of i-th category for the CTB at the location (rx, ry) for the colour component cIdx.

When sao_offset_sign[cIdx][rx][ry][i] is not present, it is inferred as follows:

– If sao_merge_left_flag is equal to 1, sao_offset_sign[cIdx][rx][ry][i] is inferred to be equal to sao_offset_sign[cIdx][rx − 1][ry][i].
– Otherwise, if sao_merge_up_flag is equal to 1, sao_offset_sign[cIdx][rx][ry][i] is inferred to be equal to sao_offset_sign[cIdx][rx][ry − 1][i].
– Otherwise, if SaoTypeIdx[cIdx][rx][ry] is equal to 2, the following applies:
  – If i is equal to 0 or 1, sao_offset_sign[cIdx][rx][ry][i] is inferred to be equal 0.
  – Otherwise (i is equal to 2 or 3), sao_offset_sign[cIdx][rx][ry][i] is inferred to be equal 1.
– Otherwise, sao_offset_sign[cIdx][rx][ry][i] is inferred to be equal 0.

The variable log2OffsetScale is derived as follows:

– If cIdx is equal to 0, log2OffsetScale is set equal to log2_sao_offset_scale_luma.

– Otherwise (cIdx is equal to 1 or 2), log2OffsetScale is set equal to log2_sao_offset_scale_chroma.

The list SaoOffsetVal[cIdx][rx][ry][i] for i ranging from 0 to 4, inclusive, is derived as follows:

```
SaoOffsetVal[ cIdx ][ rx ][ ry ][ 0 ] = 0
for( i = 0; i < 4; i++ )
SaoOffsetVal[ cIdx ][ rx ][ ry ][ i + 1 ] = ( 1 − 2 * sao_offset_sign[ cIdx ][ rx ][ ry ][ i ] ) *        (7-72)
        sao_offset_abs[ cIdx ][ rx ][ ry ][ i ] << log2OffsetScale
```

sao_band_position[cIdx][rx][ry] specifies the displacement of the band offset of the sample range when SaoTypeIdx[cIdx][rx][ry] is equal to 1.

When sao_band_position[cIdx][rx][ry] is not present, it is inferred as follows:

– If sao_merge_left_flag is equal to 1, sao_band_position[cIdx][rx][ry] is inferred to be equal to sao_band_position[cIdx][rx − 1][ry].

– Otherwise, if sao_merge_up_flag is equal to 1, sao_band_position[cIdx][rx][ry] is inferred to be equal to sao_band_position[cIdx][rx][ry − 1].

– Otherwise, sao_band_position[cIdx][rx][ry] is inferred to be equal to 0.

sao_eo_class_luma specifies the edge offset class for the luma component. The array SaoEoClass[cIdx][rx][ry] specifies the offset type as specified in Table 7-9 for the CTB at the location (rx, ry) for the colour component cIdx. The value of SaoEoClass[0][rx][ry] is derived as follows:

– If sao_eo_class_luma is present, SaoEoClass[0][rx][ry] is set equal to sao_eo_class_luma.

– Otherwise (sao_eo_class_luma is not present), SaoEoClass[0][rx][ry] is derived as follows:

  – If sao_merge_left_flag is equal to 1, SaoEoClass[0][rx][ry] is set equal to SaoEoClass[0][rx − 1][ry].

  – Otherwise, if sao_merge_up_flag is equal to 1, SaoEoClass[0][rx][ry] is set equal to SaoEoClass[0][rx][ry − 1].

  – Otherwise, SaoEoClass[0][rx][ry] is set equal to 0.

sao_eo_class_chroma specifies the edge offset class for the chroma components. The values of SaoEoClass[cIdx][rx][ry] are derived as follows for cIdx equal to 1..2:

– If sao_eo_class_chroma is present, SaoEoClass[cIdx][rx][ry] is set equal to sao_eo_class_chroma.

– Otherwise (sao_eo_class_chroma is not present), SaoEoClass[cIdx][rx][ry] is derived as follows:

  – If sao_merge_left_flag is equal to 1, SaoEoClass[cIdx][rx][ry] is set equal to SaoEoClass[cIdx][rx − 1][ry].

  – Otherwise, if sao_merge_up_flag is equal to 1, SaoEoClass[cIdx][rx][ry] is set equal to SaoEoClass[cIdx][rx][ry − 1].

  – Otherwise, SaoEoClass[cIdx][rx][ry] is set equal to 0.

Table 7-9 – Specification of the SAO edge offset class

SaoEoClass[cIdx][rx][ry]	SAO edge offset class (informative)
0	1D 0-degree edge offset
1	1D 90-degree edge offset
2	1D 135-degree edge offset
3	1D 45-degree edge offset

7.4.9.4 Coding quadtree semantics

split_cu_flag[x0][y0] specifies whether a coding unit is split into coding units with half horizontal and vertical size. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When split_cu_flag[x0][y0] is not present, the following applies:

– If log2CbSize is greater than MinCbLog2SizeY, the value of split_cu_flag[x0][y0] is inferred to be equal to 1.

– Otherwise (log2CbSize is equal to MinCbLog2SizeY), the value of split_cu_flag[x0][y0] is inferred to be equal to 0.

The array CtDepth[x][y] specifies the coding tree depth for a luma coding block covering the location (x, y). When split_cu_flag[x0][y0] is equal to 0, CtDepth[x][y] is inferred to be equal to cqtDepth for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1.

7.4.9.5 Coding unit semantics

cu_transquant_bypass_flag equal to 1 specifies that the scaling and transform process as specified in clause 8.6 and the in-loop filter process as specified in clause 8.7 are bypassed. When cu_transquant_bypass_flag is not present, it is inferred to be equal to 0.

cu_skip_flag[x0][y0] equal to 1 specifies that for the current coding unit, when decoding a P or B slice, no more syntax elements except the merging candidate index merge_idx[x0][y0] are parsed after cu_skip_flag[x0][y0]. cu_skip_flag[x0][y0] equal to 0 specifies that the coding unit is not skipped. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When cu_skip_flag[x0][y0] is not present, it is inferred to be equal to 0.

pred_mode_flag equal to 0 specifies that the current coding unit is coded in inter prediction mode. pred_mode_flag equal to 1 specifies that the current coding unit is coded in intra prediction mode. The variable CuPredMode[x][y] is derived as follows for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1:

– If pred_mode_flag is equal to 0, CuPredMode[x][y] is set equal to MODE_INTER.

– Otherwise (pred_mode_flag is equal to 1), CuPredMode[x][y] is set equal to MODE_INTRA.

When pred_mode_flag is not present, the variable CuPredMode[x][y] is derived as follows for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1:

– If slice_type is equal to I, CuPredMode[x][y] is inferred to be equal to MODE_INTRA.

– Otherwise (slice_type is equal to P or B), when cu_skip_flag[x0][y0] is equal to 1, CuPredMode[x][y] is inferred to be equal to MODE_SKIP.

palette_mode_flag[x0][y0] equal to 1 specifies that the current coding unit is coded using the palette mode. palette_mode_flag[x0][y0] equal to 0 specifies that the current coding unit is not coded using the palette mode. The array indices x0 and y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When palette_mode_flag[x0][y0] is not present, it is inferred to be equal to 0.

part_mode specifies the partitioning mode of the current coding unit. The semantics of part_mode depend on CuPredMode[x0][y0]. The variables PartMode and IntraSplitFlag are derived from the value of part_mode as defined in Table 7-10.

The value of part_mode is restricted as follows:

– If CuPredMode[x0][y0] is equal to MODE_INTRA, part_mode shall be equal to 0 or 1.

– Otherwise (CuPredMode[x0][y0] is equal to MODE_INTER), the following applies:

    – If log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 1, part_mode shall be in the range of 0 to 2, inclusive, or in the range of 4 to 7, inclusive.

    – Otherwise, if log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 0, or log2CbSize is equal to 3, part_mode shall be in the range of 0 to 2, inclusive.

    – Otherwise (log2CbSize is greater than 3 and equal to MinCbLog2SizeY), the value of part_mode shall be in the range of 0 to 3, inclusive.

When part_mode is not present, the variables PartMode and IntraSplitFlag are derived as follows:

– PartMode is set equal to PART_2Nx2N.

– IntraSplitFlag is set equal to 0.

pcm_flag[x0][y0] equal to 1 specifies that the pcm_sample( ) syntax structure is present and the transform_tree( ) syntax structure is not present in the coding unit including the luma coding block at the location (x0, y0). pcm_flag[x0][y0] equal to 0 specifies that pcm_sample( ) syntax structure is not present. When pcm_flag[x0][y0] is not present, it is inferred to be equal to 0.

The value of pcm_flag[x0 + i][y0 + j] with i = 1..nCbS − 1, j = 1..nCbS − 1 is inferred to be equal to pcm_flag[x0][y0].

pcm_alignment_zero_bit is a bit equal to 0.

Table 7-10 – Name association to prediction mode and partitioning type

CuPredMode[x0][y0]	part_mode	IntraSplitFlag	PartMode
MODE_INTRA	0	0	PART_2Nx2N
	1	1	PART_NxN
MODE_INTER	0	0	PART_2Nx2N
	1	0	PART_2NxN
	2	0	PART_Nx2N
	3	0	PART_NxN
	4	0	PART_2NxnU
	5	0	PART_2NxnD
	6	0	PART_nLx2N
	7	0	PART_nRx2N

The syntax elements **prev_intra_luma_pred_flag**[x0 + i][y0 + j], **mpm_idx**[x0 + i][y0 + j] and **rem_intra_luma_pred_mode**[x0 + i][y0 + j] specify the intra prediction mode for luma samples. The array indices x0 + i, y0 + j specify the location (x0 + i, y0 + j) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. When prev_intra_luma_pred_flag[x0 + i][y0 + j] is equal to 1, the intra prediction mode is inferred from a neighbouring intra-predicted prediction unit according to clause 8.4.2.

intra_chroma_pred_mode[x0][y0] specifies the intra prediction mode for chroma samples. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

rqt_root_cbf equal to 1 specifies that the transform_tree( ) syntax structure is present for the current coding unit. rqt_root_cbf equal to 0 specifies that the transform_tree( ) syntax structure is not present for the current coding unit.

When rqt_root_cbf is not present, its value is inferred to be equal to 1.

7.4.9.6 Prediction unit semantics

mvp_l0_flag[x0][y0] specifies the motion vector predictor index of list 0 where x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When mvp_l0_flag[x0][y0] is not present, it is inferred to be equal to 0.

mvp_l1_flag[x0][y0] has the same semantics as mvp_l0_flag, with l0 and list 0 replaced by l1 and list 1, respectively.

merge_flag[x0][y0] specifies whether the inter prediction parameters for the current prediction unit are inferred from a neighbouring inter-predicted partition. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When merge_flag[x0][y0] is not present, it is inferred as follows:

– If CuPredMode[x0][y0] is equal to MODE_SKIP, merge_flag[x0][y0] is inferred to be equal to 1.

– Otherwise, merge_flag[x0][y0] is inferred to be equal to 0.

merge_idx[x0][y0] specifies the merging candidate index of the merging candidate list where x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When merge_idx[x0][y0] is not present, it is inferred to be equal to 0.

inter_pred_idc[x0][y0] specifies whether list0, list1, or bi-prediction is used for the current prediction unit according to Table 7-11. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

Table 7-11 – Name association to inter prediction mode

inter_pred_idc	**Name of inter_pred_idc**	
	(nPbW + nPbH) != 12	(nPbW + nPbH) = = 12
0	PRED_L0	PRED_L0
1	PRED_L1	PRED_L1
2	PRED_BI	na

When inter_pred_idc[x0][y0] is not present, it is inferred to be equal to PRED_L0.

ref_idx_l0[x0][y0] specifies the list 0 reference picture index for the current prediction unit. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When ref_idx_l0[x0][y0] is not present it is inferred to be equal to 0.

ref_idx_l1[x0][y0] has the same semantics as ref_idx_l0, with l0 and list 0 replaced by l1 and list 1, respectively.

7.4.9.7 PCM sample semantics

pcm_sample_luma[i] represents a coded luma sample value in the raster scan within the coding unit. The number of bits used to represent each of these samples is PcmBitDepth_Y.

pcm_sample_chroma[i] represents a coded chroma sample value in the raster scan within the coding unit. The first half of the values represent coded Cb samples and the remaining half of the values represent coded Cr samples. The number of bits used to represent each of these samples is PcmBitDepth_C.

7.4.9.8 Transform tree semantics

split_transform_flag[x0][y0][trafoDepth] specifies whether a block is split into four blocks with half horizontal and half vertical size for the purpose of transform coding. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding block into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding blocks.

The variable interSplitFlag is derived as follows:

– If max_transform_hierarchy_depth_inter is equal to 0 and CuPredMode[x0][y0] is equal to MODE_INTER and PartMode is not equal to PART_2Nx2N and trafoDepth is equal to 0, interSplitFlag is set equal to 1.

– Otherwise, interSplitFlag is set equal to 0.

When split_transform_flag[x0][y0][trafoDepth] is not present, it is inferred as follows:

– If one or more of the following conditions are true, the value of split_transform_flag[x0][y0][trafoDepth] is inferred to be equal to 1:
  – log2TrafoSize is greater than MaxTbLog2SizeY.
  – IntraSplitFlag is equal to 1 and trafoDepth is equal to 0.
  – interSplitFlag is equal to 1.

– Otherwise, the value of split_transform_flag[x0][y0][trafoDepth] is inferred to be equal to 0.

cbf_luma[x0][y0][trafoDepth] equal to 1 specifies that the luma transform block contains one or more transform coefficient levels not equal to 0. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the

current subdivision level of a coding block into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding blocks.

When cbf_luma[x0][y0][trafoDepth] is not present, it is inferred to be equal to 1.

cbf_cb[x0][y0][trafoDepth] equal to 1 specifies that the Cb transform block contains one or more transform coefficient levels not equal to 0. The array indices x0, y0 specify the top-left location (x0, y0) of the considered transform block. The array index trafoDepth specifies the current subdivision level of a coding block into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding blocks.

When cbf_cb[x0][y0][trafoDepth] is not present, it is inferred to be equal to 0.

cbf_cr[x0][y0][trafoDepth] equal to 1 specifies that the Cr transform block contains one or more transform coefficient levels not equal to 0. The array indices x0, y0 specify the top-left location (x0, y0) of the considered transform block. The array index trafoDepth specifies the current subdivision level of a coding block into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding blocks.

When cbf_cr[x0][y0][trafoDepth] is not present, it is inferred to be equal to 0.

7.4.9.9 Motion vector difference semantics

abs_mvd_greater0_flag[compIdx] specifies whether the absolute value of a motion vector component difference is greater than 0.

abs_mvd_greater1_flag[compIdx] specifies whether the absolute value of a motion vector component difference is greater than 1.

When abs_mvd_greater1_flag[compIdx] is not present, it is inferred to be equal to 0.

abs_mvd_minus2[compIdx] plus 2 specifies the absolute value of a motion vector component difference.

When abs_mvd_minus2[compIdx] is not present, it is inferred to be equal to −1.

mvd_sign_flag[compIdx] specifies the sign of a motion vector component difference as follows:

– If mvd_sign_flag[compIdx] is equal to 0, the corresponding motion vector component difference has a positive value.

– Otherwise (mvd_sign_flag[compIdx] is equal to 1), the corresponding motion vector component difference has a negative value.

When mvd_sign_flag[compIdx] is not present, it is inferred to be equal to 0.

The motion vector difference lMvd[compIdx] for compIdx = 0..1 is derived as follows:

$$\begin{aligned}&\text{lMvd[compIdx]} = \text{abs\_mvd\_greater0\_flag[compIdx]} * \\&\quad (\text{ abs\_mvd\_minus2[compIdx]} + 2\ ) * (\ 1 - 2 * \text{mvd\_sign\_flag[compIdx]}\ )\end{aligned} \tag{7-73}$$

The variable MvdLX[x0][y0][compIdx], with X being 0 or 1, specifies the difference between a list X vector component to be used and its prediction. The value of MvdLX[x0][y0][compIdx] shall be in the range of -2^{15} to $2^{15} - 1$, inclusive. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

– If refList is equal to 0, MvdL0[x0][y0][compIdx] is set equal to lMvd[compIdx] for compIdx = 0..1.

– Otherwise (refList is equal to 1), MvdL1[x0][y0][compIdx] is set equal to lMvd[compIdx] for compIdx = 0..1.

7.4.9.10 Transform unit semantics

The transform coefficient levels are represented by the arrays TransCoeffLevel[x0][y0][cIdx][xC][yC], which are either specified in clause 7.3.8.11 or inferred as follows. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index cIdx specifies an indicator for the colour component; it is equal to 0 for Y, 1 for Cb, and 2 for Cr. The array indices xC and yC specify the transform coefficient location (xC, yC) within the current transform block. When the value of TransCoeffLevel[x0][y0][cIdx][xC][yC] is not specified in clause 7.3.8.11, it is inferred to be equal to 0.

tu_residual_act_flag[x0][y0] equal to 1 specifies that adaptive colour transform is applied to the residual samples of the current transform unit. tu_residual_act_flag[x0][y0] equal to 0 specifies that adaptive colour transform is not applied to the residual samples of the current transform unit. When tu_residual_act_flag[x0][y0] is not present, it is inferred to be equal to 0.

When cu_transquant_bypass_flag is equal to 1 and bit_depth_luma_minus8 is not equal to bit_depth_chroma_minus8, the value of tu_residual_act_flag[x0][y0], when present, shall be equal to 0.

7.4.9.11 Residual coding semantics

For intra prediction, different scanning orders are used. The variable scanIdx specifies which scan order is used where scanIdx equal to 0 specifies an up-right diagonal scan order, scanIdx equal to 1 specifies a horizontal scan order, and scanIdx equal to 2 specifies a vertical scan order. The value of scanIdx is derived as follows:

– If CuPredMode[x0][y0] is equal to MODE_INTRA and one or more of the following conditions are true:
  – log2TrafoSize is equal to 2.
  – log2TrafoSize is equal to 3 and cIdx is equal to 0.
  – log2TrafoSize is equal to 3 and ChromaArrayType is equal to 3.

  predModeIntra is derived as follows:
  – If cIdx is equal to 0, predModeIntra is set equal to IntraPredModeY[x0][y0].
  – Otherwise, predModeIntra is set equal to IntraPredModeC.

  scanIdx is derived as follows:
  – If predModeIntra is in the range of 6 to 14, inclusive, scanIdx is set equal to 2.
  – Otherwise if predModeIntra is in the range of 22 to 30, inclusive, scanIdx is set equal to 1.
  – Otherwise, scanIdx is set equal to 0.
– Otherwise, scanIdx is set equal to 0.

transform_skip_flag[x0][y0][cIdx] specifies whether a transform is applied to the associated transform block or not: The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index cIdx specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb and equal to 2 for Cr. transform_skip_flag[x0][y0][cIdx] equal to 1 specifies that no transform is applied to the current transform block. transform_skip_flag[x0][y0][cIdx] equal to 0 specifies that the decision whether transform is applied to the current transform block or not depends on other syntax elements. When transform_skip_flag[x0][y0][cIdx] is not present, it is inferred to be equal to 0.

explicit_rdpcm_flag[x0][y0][cIdx] specifies whether the residual modification process for blocks using a transform bypass is applied to the associated transform block or not. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index cIdx specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb and equal to 2 for Cr. explicit_rdpcm_flag[x0][y0][cIdx] equal to 1 specifies that the residual modification process is applied to the current transform block. explicit_rdpcm_flag[x0][y0][cIdx] equal to 0 specifies that no residual modification process is applied to the current transform block. When explicit_rdpcm_flag[x0][y0][cIdx] is not present, it is inferred to be equal to 0.

explicit_rdpcm_dir_flag[x0][y0][cIdx] specifies the direction to be used by the residual modification process for the associated transform block. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index cIdx specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb and equal to 2 for Cr.

last_sig_coeff_x_prefix specifies the prefix of the column position of the last significant coefficient in scanning order within a transform block. The values of last_sig_coeff_x_prefix shall be in the range of 0 to (log2TrafoSize << 1) − 1, inclusive.

last_sig_coeff_y_prefix specifies the prefix of the row position of the last significant coefficient in scanning order within a transform block. The values of last_sig_coeff_y_prefix shall be in the range of 0 to (log2TrafoSize << 1) − 1, inclusive.

last_sig_coeff_x_suffix specifies the suffix of the column position of the last significant coefficient in scanning order within a transform block. The values of last_sig_coeff_x_suffix shall be in the range of 0 to (1 << ((last_sig_coeff_x_prefix >> 1) − 1)) − 1, inclusive.

The column position of the last significant coefficient in scanning order within a transform block LastSignificantCoeffX is derived as follows:

– If last_sig_coeff_x_suffix is not present, the following applies:

LastSignificantCoeffX = last_sig_coeff_x_prefix (7-74)

– Otherwise (last_sig_coeff_x_suffix is present), the following applies:

$$\text{LastSignificantCoeffX} = (1 \ll ((\text{last\_sig\_coeff\_x\_prefix} \gg 1) - 1)) * (2 + (\text{last\_sig\_coeff\_x\_prefix} \,\&\, 1)) + \text{last\_sig\_coeff\_x\_suffix} \quad (7\text{-}75)$$

last_sig_coeff_y_suffix specifies the suffix of the row position of the last significant coefficient in scanning order within a transform block. The values of last_sig_coeff_y_suffix shall be in the range of 0 to $(1 \ll ((\text{last\_sig\_coeff\_y\_prefix} \gg 1) - 1)) - 1$, inclusive.

The row position of the last significant coefficient in scanning order within a transform block LastSignificantCoeffY is derived as follows:

– If last_sig_coeff_y_suffix is not present, the following applies:

$$\text{LastSignificantCoeffY} = \text{last\_sig\_coeff\_y\_prefix} \quad (7\text{-}76)$$

– Otherwise (last_sig_coeff_y_suffix is present), the following applies:

$$\text{LastSignificantCoeffY} = (1 \ll ((\text{last\_sig\_coeff\_y\_prefix} \gg 1) - 1)) * (2 + (\text{last\_sig\_coeff\_y\_prefix} \,\&\, 1)) + \text{last\_sig\_coeff\_y\_suffix} \quad (7\text{-}77)$$

When scanIdx is equal to 2, the coordinates are swapped as follows:

$$(\text{LastSignificantCoeffX}, \text{LastSignificantCoeffY}) = \text{Swap}(\text{LastSignificantCoeffX}, \text{LastSignificantCoeffY}) \quad (7\text{-}78)$$

coded_sub_block_flag[xS][yS] specifies the following for the sub-block at location (xS, yS) within the current transform block, where a sub-block is a (4x4) array of 16 transform coefficient levels:

– If coded_sub_block_flag[xS][yS] is equal to 0, the 16 transform coefficient levels of the sub-block at location (xS, yS) are inferred to be equal to 0.

– Otherwise (coded_sub_block_flag[xS][yS] is equal to 1), the following applies:

  – If (xS, yS) is equal to (0, 0) and (LastSignificantCoeffX, LastSignificantCoeffY) is not equal to (0, 0), at least one of the 16 sig_coeff_flag syntax elements is present for the sub-block at location (xS, yS).

  – Otherwise, at least one of the 16 transform coefficient levels of the sub-block at location (xS, yS) has a non-zero value.

When coded_sub_block_flag[xS][yS] is not present, it is inferred as follows:

– If one or more of the following conditions are true, coded_sub_block_flag[xS][yS] is inferred to be equal to 1:

  – (xS, yS) is equal to (0, 0).

  – (xS, yS) is equal to (LastSignificantCoeffX >> 2, LastSignificantCoeffY >> 2).

– Otherwise, coded_sub_block_flag[xS][yS] is inferred to be equal to 0.

sig_coeff_flag[xC][yC] specifies for the transform coefficient location (xC, yC) within the current transform block whether the corresponding transform coefficient level at the location (xC, yC) is non-zero as follows:

– If sig_coeff_flag[xC][yC] is equal to 0, the transform coefficient level at the location (xC, yC) is set equal to 0.

– Otherwise (sig_coeff_flag[xC][yC] is equal to 1), the transform coefficient level at the location (xC, yC) has a non-zero value.

When sig_coeff_flag[xC][yC] is not present, it is inferred as follows:

– If (xC, yC) is the last significant location (LastSignificantCoeffX, LastSignificantCoeffY) in scan order or all of the following conditions are true, sig_coeff_flag[xC][yC] is inferred to be equal to 1:

  – (xC & 3, yC & 3) is equal to (0, 0).

  – inferSbDcSigCoeffFlag is equal to 1.

  – coded_sub_block_flag[xS][yS] is equal to 1.

– Otherwise, sig_coeff_flag[xC][yC] is inferred to be equal to 0.

coeff_abs_level_greater1_flag[n] specifies for the scanning position n whether there are absolute values of transform coefficient levels greater than 1.

When coeff_abs_level_greater1_flag[n] is not present, it is inferred to be equal to 0.

coeff_abs_level_greater2_flag[n] specifies for the scanning position n whether there are absolute values of transform coefficient levels greater than 2.

When coeff_abs_level_greater2_flag[n] is not present, it is inferred to be equal to 0.

coeff_sign_flag[n] specifies the sign of a transform coefficient level for the scanning position n as follows:

– If coeff_sign_flag[n] is equal to 0, the corresponding transform coefficient level has a positive value.

– Otherwise (coeff_sign_flag[n] is equal to 1), the corresponding transform coefficient level has a negative value.

When coeff_sign_flag[n] is not present, it is inferred to be equal to 0.

coeff_abs_level_remaining[n] is the remaining absolute value of a transform coefficient level that is coded with Golomb-Rice code at the scanning position n. When coeff_abs_level_remaining[n] is not present, it is inferred to be equal to 0.

It is a requirement of bitstream conformance that the value of coeff_abs_level_remaining[n] shall be constrained such that the corresponding value of TransCoeffLevel[x0][y0][cIdx][xC][yC] is in the range of CoeffMin_Y to CoeffMax_Y, inclusive, for cIdx equal to 0 and in the range of CoeffMin_C to CoeffMax_C, inclusive, for cIdx not equal to 0.

7.4.9.12 Cross-component prediction semantics

log2_res_scale_abs_plus1[c] minus 1 specifies the base 2 logarithm of the magnitude of the scaling factor ResScaleVal used in cross-component residual prediction. When not present, log2_res_scale_abs_plus1 is inferred to be equal to 0.

res_scale_sign_flag[c] specifies the sign of the scaling factor used in cross-component residual prediction as follows:

– If res_scale_sign_flag[c] is equal to 0, the corresponding ResScaleVal has a positive value.

– Otherwise (res_scale_sign_flag[c] is equal to 1), the corresponding ResScaleVal has a negative value.

The variable ResScaleVal[cIdx][x0][y0] specifies the scaling factor used in cross-component residual prediction. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index cIdx specifies an indicator for the colour component; it is equal to 1 for Cb and equal to 2 for Cr.

The variable ResScaleVal[cIdx][x0][y0] is derived as follows:

– If log2_res_scale_abs_plus1[cIdx − 1] is equal to 0, the following applies:

$$\text{ResScaleVal}[\ \text{cIdx}\ ][\ \text{x0}\ ][\ \text{y0}\ ] = 0 \tag{7-79}$$

– Otherwise (log2_res_scale_abs_plus1[cIdx − 1] is not equal to 0), the following applies:

$$\text{ResScaleVal}[\ \text{cIdx}\ ][\ \text{x0}\ ][\ \text{y0}\ ] = (\ 1 \ll (\ \text{log2\_res\_scale\_abs\_plus1}[\ \text{cIdx} - 1\ ] - 1\ )\ ) * (\ 1 - 2 * \text{res\_scale\_sign\_flag}[\ \text{cIdx} - 1\ ]\ ) \tag{7-80}$$

7.4.9.13 Palette semantics

In the following semantics, the array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

The predictor palette consists of palette entries from previous coding units that are used to predict the entries in the current palette.

The variable PredictorPaletteSize specifies the size of the predictor palette. PredictorPaletteSize is derived as specified in clause 8.4.4.2.7.

The variable PalettePredictorEntryReuseFlags[i] equal to 1 specifies that the i-th entry in the predictor palette is reused in the current palette. PalettePredictorEntryReuseFlags[i] equal to 0 specifies that the i-th entry in the predictor palette is not an entry in the current palette. All elements of the array PalettePredictorEntryReuseFlags[i] are initialized to 0.

palette_predictor_run is used to determine the number of zeros that precede a non-zero entry in the array PalettePredictorEntryReuseFlags.

It is a requirement of bitstream conformance that the value of palette_predictor_run shall be in the range of 0 to (PredictorPaletteSize − predictorEntryIdx), inclusive, where predictorEntryIdx corresponds to the current position in the

array PalettePredictorEntryReuseFlags. The variable NumPredictedPaletteEntries specifies the number of entries in the current palette that are reused from the predictor palette. The value of NumPredictedPaletteEntries shall be in the range of 0 to palette_max_size, inclusive.

num_signalled_palette_entries specifies the number of entries in the current palette that are explicitly signalled.

When num_signalled_palette_entries is not present, it is inferred to be equal to 0.

The variable CurrentPaletteSize specifies the size of the current palette and is derived as follows:

CurrentPaletteSize = NumPredictedPaletteEntries + num_signalled_palette_entries (7-81)

The value of CurrentPaletteSize shall be in the range of 0 to palette_max_size, inclusive.

new_palette_entries[cIdx][i] specifies the value for the i-th signalled palette entry for the colour component cIdx.

The variable PredictorPaletteEntries[cIdx][i] specifies the i-th element in the predictor palette for the colour component cIdx.

The variable CurrentPaletteEntries[cIdx][i] specifies the i-th element in the current palette for the colour component cIdx and is derived as follows:

```
numComps = ( ChromaArrayType  = =  0 ) ? 1 : 3
numPredictedPaletteEntries = 0
for( i = 0; i < PredictorPaletteSize; i++ )
    if( PalettePredictorEntryReuseFlags[ i ] ) {
        for( cIdx = 0; cIdx < numComps; cIdx++ )
            CurrentPaletteEntries[ cIdx ][ numPredictedPaletteEntries ] =
PredictorPaletteEntries[ cIdx ][ i ]
        numPredictedPaletteEntries++
    }
for( cIdx = 0; cIdx < numComps; cIdx++ )                                    (7-82)
    for( i = 0; i < num_signalled_palette_entries; i++ )
        CurrentPaletteEntries[ cIdx ][ numPredictedPaletteEntries + i ] = new_palette_entries[ cIdx ][ i ]
```

palette_escape_val_present_flag equal to 1 specifies that the current coding unit contains at least one escape-coded sample. escape_val_present_flag equal to 0 specifies that there are no escape-coded samples in the current coding unit. When not present, the value of palette_escape_val_present_flag is inferred to be equal to 1.

The variable MaxPaletteIndex specifies the maximum possible value for a palette index for the current coding unit. The value of MaxPaletteIndex is set equal to CurrentPaletteSize − 1 + palette_escape_val_present_flag.

num_palette_indices_minus1 plus 1 is the number of palette indices explicitly signalled or inferred for the current block.

When num_palette_indices_minus1 is not present, it is inferred to be equal to 0.

palette_idx_idc is an indication of an index to the array represented by CurrentPaletteEntries. The value of palette_idx_idc shall be in the range of 0 to MaxPaletteIndex, inclusive, for the first index in the block and in the range of 0 to (MaxPaletteIndex − 1), inclusive, for the remaining indices in the block.

When palette_idx_idc is not present, it is inferred to be equal to 0.

The variable PaletteIndexIdc[i] stores the i-th palette_idx_idc explicitly signalled or inferred. All elements of the array PaletteIndexIdc[i] are initialized to 0.

copy_above_indices_for_final_run_flag equal to 1 specifies that the palette indices of the last positions in the coding unit are copied from the palette indices in the row above. copy_above_indices_for_final_run_flag equal to 0 specifies that the palette indices of the last positions in the coding unit are copied from PaletteIndexIdc[num_palette_indices_minus1].

When copy_above_indices_for_final_run_flag is not present, it is inferred to be equal to 0.

palette_transpose_flag equal to 1 specifies that the transpose process is applied to the reconstructed sample array at the output of palette mode decoding as specified in clause 8.4.4.2.7 in the current coding unit. palette_transpose_flag equal to 0 specifies that the transpose process is not applied to the reconstructed sample array at the output of palette mode decoding clause 8.4.4.2.7 in the current coding unit. When not present, the value of palette_transpose_flag is inferred to be equal to 0.

copy_above_palette_indices_flag equal to 1 specifies that the palette index is equal to the palette index at the same location in the row above. copy_above_palette_indices_flag equal to 0 specifies that an indication of the palette index of the sample is coded in the bitstream or inferred.

The variable CopyAboveIndicesFlag[xC][yC] equal to 1 specifies that the palette index is copied from the palette index in the row above. CopyAboveIndicesFlag[xC][yC] equal to 0 specifies that the palette index is explicitly coded in the bitstream or inferred. The array indices xC, yC specify the location (xC, yC) of the sample relative to the top-left luma sample of the picture.

The variable PaletteIndexMap[xC][yC] specifies a palette index, which is an index to the array represented by CurrentPaletteEntries. The array indices xC, yC specify the location (xC, yC) of the sample relative to the top-left luma sample of the picture. The value of PaletteIndexMap[xC][yC] shall be in the range of 0 to MaxPaletteIndex, inclusive.

The variable adjustedRefPaletteIndex is derived as follows:

```
adjustedRefPaletteIndex = MaxPaletteIndex + 1
log2BlkSize = Log2( nCbs ) − 2
if( PaletteScanPos > 0 ) {
    xcPrev = x0 + ScanOrder[ log2BlkSize ][ 3 ][ PaletteScanPos − 1 ][ 0 ]
    ycPrev = y0 + ScanOrder[ log2BlkSize ][ 3 ][ PaletteScanPos − 1 ][ 1 ]
    if( CopyAboveIndicesFlag[ xcPrev ][ ycPrev ]  = =  0 )
            adjustedRefPaletteIndex = PaletteIndexMap[ xcPrev ][ ycPrev ]          (7-83)
    else
            adjustedRefPaletteIndex = PaletteIndexMap[ xC ][ yC − 1 ]
```

When CopyAboveIndicesFlag[xC][yC] is equal to 0, the variable CurrPaletteIndex is derived as follows:

```
if( CurrPaletteIndex  >=  adjustedRefPaletteIndex )
    CurrPaletteIndex++                                                       (7-84)
```

palette_run_prefix, when present, is used in the derivation of the variable PaletteRunMinus1.

palette_run_suffix is used in the derivation of the variable PaletteRunMinus1. When not present, the value of palette_run_suffix is inferred to be equal to 0.

When RunToEnd is equal to 0, the variable PaletteRunMinus1 is derived as follows:

– If PaletteMaxRunMinus1 is equal to 0, PaletteRunMinus1 is set equal to 0.

– Otherwise (PaletteMaxRunMinus1 is greater than 0) the following applies:

  – If palette_run_prefix is less than 2, the following applies:

```
PaletteRunMinus1 = palette_run_prefix
    (7-85)
```

– Otherwise (palette_run_prefix is greater than or equal to 2), the following applies:

$$\text{PrefixOffset} = 1 \ll (\text{palette\_run\_prefix} - 1)$$
$$\text{PaletteRunMinus1} = \text{PrefixOffset} + \text{palette\_run\_suffix} \quad (7\text{-}86)$$

The variable PaletteRunMinus1 is used as follows:

– If CopyAboveIndicesFlag[xC][yC] is equal to 0, PaletteRunMinus1 specifies the number of consecutive locations minus 1 with the same palette index.

– Otherwise, PaletteRunMinus1 specifies the number of consecutive locations minus 1 with the same palette index as used in the corresponding position in the row above.

When RunToEnd is equal to 0, the variable PaletteMaxRunMinus1 represents the maximum possible value for PaletteRunMinus1 and it is a requirement of bitstream conformance that the value of PaletteMaxRunMinus1 shall be greater than or equal to 0.

palette_escape_val specifies the quantized escape-coded sample value for a component.

The variable PaletteEscapeVal[cIdx][xC][yC] specifies the escape value of a sample for which PaletteIndexMap[xC][yC] is equal to MaxPaletteIndex and palette_escape_val_present_flag is equal to 1. The array index cIdx specifies the colour component. The array indices xC, yC specify the location (xC, yC) of the sample relative to the top-left luma sample of the picture.

It is a requirement of bitstream conformance that PaletteEscapeVal[cIdx][xC][yC] shall be in the range of 0 to $(1 \ll (\text{BitDepth}_Y + 1)) - 1$, inclusive, for cIdx equal to 0, and in the range of 0 to $(1 \ll (\text{BitDepth}_C + 1)) - 1$, inclusive, for cIdx not equal to 0.

7.4.9.14 Delta QP semantics

cu_qp_delta_abs, when present, specifies the absolute value of the difference CuQpDeltaVal between the luma quantization parameter of the current coding unit and its prediction.

cu_qp_delta_sign_flag, when present, specifies the sign of the difference CuQpDeltaVal between the luma quantization parameter of the current coding unit and its prediction.

It is a requirement of bitstream conformance that the value of CuQpDeltaVal shall be in the range of −(26 + QpBdOffsetY / 2) to +(25 + QpBdOffsetY / 2), inclusive.

7.4.9.15 Chroma QP offset semantics

cu_chroma_qp_offset_flag, when present and equal to 1, specifies that an entry in the cb_qp_offset_list[ ] is used to determine the value of CuQpOffsetCb and a corresponding entry in the cr_qp_offset_list[ ] is used to determine the value of CuQpOffsetCr. cu_chroma_qp_offset_flag equal to 0 specifies that these lists are not used to determine the values of CuQpOffsetCb and CuQpOffsetCr.

cu_chroma_qp_offset_idx, when present, specifies the index into the cb_qp_offset_list[ ] and cr_qp_offset_list[ ] that is used to determine the value of CuQpOffsetCb and CuQpOffsetCr. When present, the value of cu_chroma_qp_offset_idx shall be in the range of 0 to chroma_qp_offset_list_len_minus1, inclusive. When not present, the value of cu_chroma_qp_offset_idx is inferred to be equal to 0.

When cu_chroma_qp_offset_flag is present, the following applies:

– The variable IsCuChromaQpOffsetCoded is set equal to 1.

– The variables CuQpOffsetCb and CuQpOffsetCr are derived as follows:

  – If cu_chroma_qp_offset_flag is equal to 1, the following applies:

$$\text{CuQpOffsetCb} = \text{cb\_qp\_offset\_list}[\text{cu\_chroma\_qp\_offset\_idx}] \quad (7\text{-}87)$$

$$\text{CuQpOffsetCr} = \text{cr\_qp\_offset\_list}[\text{cu\_chroma\_qp\_offset\_idx}] \quad (7\text{-}88)$$

  – Otherwise (cu_chroma_qp_offset_flag is equal to 0), CuQpOffsetCb and CuQpOffsetCr are both set equal to 0.

NOTE – When cu_chroma_qp_offset_enabled_flag is equal to 0, CuQpOffsetCb and CuQpOffsetCr are not modified after being initialized to 0 for the slice as specified in clause 7.4.7.1.

8 Decoding process

8.1 General decoding process

8.1.1 General

Input to this process is a bitstream. Output of this process is a list of decoded pictures.

The decoding process is specified such that all decoders that conform to a specified profile, tier and level will produce numerically identical cropped decoded output pictures when invoking the decoding process associated with that profile for a bitstream conforming to that profile, tier and level. Any decoding process that produces identical cropped decoded output pictures to those produced by the process described herein (with the correct output order or output timing, as specified) conforms to the decoding process requirements of this Specification.

NOTE 1 – For the purpose of best-effort decoding, a decoder that conforms to a particular profile at a given tier and level may additionally decode some bitstreams conforming to a different tier, level or profile without necessarily using a decoding process that produces numerically identical cropped decoded output pictures to those produced by the process specified herein (without claiming conformance to the other profile, tier and level).

At the beginning of decoding a coded video sequence group (CVSG), after activating the VPS RBSP that is active for the entire CVSG and before decoding any VCL NAL units of the CVSG, clause 8.1.2 is invoked with the CVSG as input.

8.1.2 CVSG decoding process

Input to this process is a CVSG. Output of this process is a list of decoded pictures.

The layer identifier list TargetDecLayerIdList, which specifies the list of nuh_layer_id values, in increasing order of nuh_layer_id values, of the NAL units to be decoded, is specified as follows:

– If some external means, not specified in this Specification, is available to set TargetDecLayerIdList, TargetDecLayerIdList is set by the external means.

– Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause C.1, TargetDecLayerIdList is set as specified in clause C.1.

– Otherwise, TargetDecLayerIdList contains only one nuh_layer_id value that is equal to 0.

The variable HighestTid, which identifies the highest temporal sub-layer to be decoded, is specified as follows:

– If some external means, not specified in this Specification, is available to set HighestTid, HighestTid is set by the external means.

– Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause C.1, HighestTid is set as specified in clause C.1.

– Otherwise, HighestTid is set equal to sps_max_sub_layers_minus1.

The variable SubPicHrdFlag is specified as follows:

– If the decoding process is invoked in a bitstream conformance test as specified in clause C.1, SubPicHrdFlag is set as specified in clause C.1.

– Otherwise, SubPicHrdFlag is set equal to (SubPicHrdPreferredFlag && sub_pic_hrd_params_present_flag).

The sub-bitstream extraction process as specified in clause 10 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.

Clause 8.1.3 is repeatedly invoked for each coded picture with nuh_layer_id equal to 0 in BitstreamToDecode in decoding order.

8.1.3 Decoding process for a coded picture with nuh_layer_id equal to 0

The decoding processes specified in this clause apply to each coded picture with nuh_layer_id equal to 0, referred to as the current picture and denoted by the variable CurrPic, in BitstreamToDecode.

Depending on the value of chroma_format_idc, the number of sample arrays of the current picture is as follows:

– If chroma_format_idc is equal to 0, the current picture consists of 1 sample array S_L.

– Otherwise (chroma_format_idc is not equal to 0), the current picture consists of 3 sample arrays S_L, S_{Cb}, S_{Cr}.

The decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause 7. When interpreting the semantics of each syntax element in each NAL unit, the term "the bitstream" (or part thereof, e.g., a CVS of the bitstream) refers to BitstreamToDecode (or part thereof).

When the current picture is a BLA picture that has nal_unit_type equal to BLA_W_LP or is a CRA picture, the following applies:

– If some external means not specified in this Specification is available to set the variable UseAltCpbParamsFlag to a value, UseAltCpbParamsFlag is set equal to the value provided by the external means.

– Otherwise, the value of UseAltCpbParamsFlag is set equal to 0.

When the current picture is an IRAP picture, the following applies:

– If the current picture is an IDR picture, a BLA picture, the first picture in the bitstream in decoding order, or the first picture that follows an end of sequence NAL unit in decoding order, the variable NoRaslOutputFlag is set equal to 1.

– Otherwise, if some external means not specified in this Specification is available to set the variable HandleCraAsBlaFlag to a value for the current picture, the variable HandleCraAsBlaFlag is set equal to the value provided by the external means and the variable NoRaslOutputFlag is set equal to HandleCraAsBlaFlag.

– Otherwise, the variable HandleCraAsBlaFlag is set equal to 0 and the variable NoRaslOutputFlag is set equal to 0.

Depending on the value of separate_colour_plane_flag, the decoding process is structured as follows:

– If separate_colour_plane_flag is equal to 0, the decoding process is invoked a single time with the current picture being the output.

– Otherwise (separate_colour_plane_flag is equal to 1), the decoding process is invoked three times. Inputs to the decoding process are all NAL units of the coded picture with identical value of colour_plane_id. The decoding process of NAL units with a particular value of colour_plane_id is specified as if only a CVS with monochrome colour format with that particular value of colour_plane_id would be present in the bitstream. The output of each of the three decoding processes is assigned to one of the 3 sample arrays of the current picture, with the NAL units with colour_plane_id equal to 0, 1 and 2 being assigned to S_L, S_{Cb} and S_{Cr}, respectively.

NOTE 1 – The variable ChromaArrayType is derived as equal to 0 when separate_colour_plane_flag is equal to 1 and chroma_format_idc is equal to 3. In the decoding process, the value of this variable is evaluated resulting in operations identical to that of monochrome pictures (when chroma_format_idc is equal to 0).

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in clause 8.2.

2. The processes in clause 8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:

   – Variables and functions relating to picture order count are derived as specified in clause 8.3.1. This needs to be invoked only for the first slice segment of a picture.

   – The decoding process for RPS in clause 8.3.2 is invoked, wherein reference pictures may be marked as "unused for reference" or "used for long-term reference". This needs to be invoked only for the first slice segment of a picture.

   – A picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture after the invocation of the in-loop filter process as specified in clause 8.7. This version of the current decoded picture is referred to as the current decoded picture after the invocation of the in-loop filter process. When TwoVersionsOfCurrDecPicFlag is equal to 0 and pps_curr_pic_ref_enabled_flag is equal to 1, this picture storage buffer is marked as "used for long-term reference". When TwoVersionsOfCurrDecPicFlag is equal to 1, another picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture immediately before the invocation of the in-loop filter process as specified in clause 8.7, and is marked as "used for long-term reference". This version of the current decoded picture is referred to as the current decoded picture before the invocation of the in-loop filter process. This needs to be invoked only for the first slice segment of a picture.

     NOTE 2 – When TwoVersionsOfCurrDecPicFlag is equal to 0, there is only one version of the current decoded picture. In this case, if pps_curr_pic_ref_enabled_flag is equal to 1, the current decoded picture is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture, otherwise it is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture. When TwoVersionsOfCurrDecPicFlag is equal to 1, there are two versions of the current decoded picture, one of which is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "unused for reference" at the end of the decoding of the current picture, and the

other version is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture.

- When the current picture is a BLA picture or is a CRA picture with NoRaslOutputFlag equal to 1, the decoding process for generating unavailable reference pictures specified in clause 8.3.3 is invoked, which needs to be invoked only for the first slice segment of a picture.
- PicOutputFlag is set as follows:
  - If the current picture is a RASL picture and NoRaslOutputFlag of the associated IRAP picture is equal to 1, PicOutputFlag is set equal to 0.
  - Otherwise, PicOutputFlag is set equal to pic_output_flag.
- At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause 8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0) and, when decoding a B slice, reference picture list 1 (RefPicList1), and the decoding process for collocated picture and no backward prediction flag specified in clause 8.3.5 is invoked for derivation of the variables ColPic and NoBackwardPredFlag.

3. The processes in clauses 8.4, 8.5, 8.6 and 8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments and the division of the slice segments into CTUs each forms a partitioning of the picture.
4. After all slices of the current picture have been decoded, the current decoded picture after the invocation of the in-loop filter process as specified in clause 8.7 is marked as "used for short-term reference". When TwoVersionsOfCurrDecPicFlag is equal to 1, the current decoded picture before the invocation of the in-loop filter process as specified in clause 8.7 is marked as "unused for reference".

8.2 NAL unit decoding process

Inputs to this process are NAL units of the current picture and their associated non-VCL NAL units.

Outputs of this process are the parsed RBSP syntax structures encapsulated within the NAL units.

The decoding process for each NAL unit extracts the RBSP syntax structure from the NAL unit and then parses the RBSP syntax structure.

8.3 Slice decoding process

8.3.1 Decoding process for picture order count

Output of this process is PicOrderCntVal, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction, and for decoder conformance checking (see clause C.5).

Each coded picture is associated with a picture order count variable, denoted as PicOrderCntVal.

When the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1, the variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows:

- Let prevTid0Pic be the previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL or SLNR picture.
- The variable prevPicOrderCntLsb is set equal to slice_pic_order_cnt_lsb of prevTid0Pic.
- The variable prevPicOrderCntMsb is set equal to PicOrderCntMsb of prevTid0Pic.

The variable PicOrderCntMsb of the current picture is derived as follows:

- If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, PicOrderCntMsb is set equal to 0.
- Otherwise, PicOrderCntMsb is derived as follows:

```
if( ( slice_pic_order_cnt_lsb < prevPicOrderCntLsb ) &&
        ( ( prevPicOrderCntLsb − slice_pic_order_cnt_lsb ) >= ( MaxPicOrderCntLsb / 2 ) ) )
    PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb                                  (8-1)
else if( (slice_pic_order_cnt_lsb > prevPicOrderCntLsb ) &&
        ( ( slice_pic_order_cnt_lsb − prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) ) )
```

```
        PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb
    else
        PicOrderCntMsb = prevPicOrderCntMsb
```

PicOrderCntVal is derived as follows:

$$\text{PicOrderCntVal} = \text{PicOrderCntMsb} + \text{slice\_pic\_order\_cnt\_lsb} \tag{8-2}$$

NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since slice_pic_order_cnt_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0.

The value of PicOrderCntVal shall be in the range of -2^{31} to $2^{31} - 1$, inclusive. In one CVS, the PicOrderCntVal values for any two coded pictures shall not be the same.

The function PicOrderCnt(picX) is specified as follows:

$$\text{PicOrderCnt(picX)} = \text{PicOrderCntVal of the picture picX} \tag{8-3}$$

The function DiffPicOrderCnt(picA, picB) is specified as follows:

$$\text{DiffPicOrderCnt(picA, picB)} = \text{PicOrderCnt(picA)} - \text{PicOrderCnt(picB)} \tag{8-4}$$

The bitstream shall not contain data that result in values of DiffPicOrderCnt(picA, picB) used in the decoding process that are not in the range of -2^{15} to $2^{15} - 1$, inclusive.

NOTE 2 – Let X be the current picture and Y and Z be two other pictures in the same CVS, Y and Z are considered to be in the same output order direction from X when both DiffPicOrderCnt(X, Y) and DiffPicOrderCnt(X, Z) are positive or both are negative.

8.3.2 Decoding process for reference picture set

This process is invoked once per picture, after decoding of a slice header but prior to the decoding of any coding unit and prior to the decoding process for reference picture list construction for the slice as specified in clause 8.3.4. This process may result in one or more reference pictures in the DPB being marked as "unused for reference" or "used for long-term reference".

NOTE 1 – The RPS is an absolute description of the reference pictures used in the decoding process of the current and future coded pictures. The RPS signalling is explicit in the sense that all reference pictures included in the RPS are listed explicitly.

A decoded picture in the DPB can be marked as "unused for reference", "used for short-term reference" or "used for long-term reference", but only one among these three at any given moment during the operation of the decoding process. Assigning one of these markings to a picture implicitly removes another of these markings when applicable. When a picture is referred to as being marked as "used for reference", this collectively refers to the picture being marked as "used for short-term reference" or "used for long-term reference" (but not both).

The variable currPicLayerId is set equal to nuh_layer_id of the current picture.

When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, all reference pictures with nuh_layer_id equal to currPicLayerId currently in the DPB (if any) are marked as "unused for reference".

Short-term reference pictures are identified by their PicOrderCntVal values. Long-term reference pictures are identified either by their PicOrderCntVal values or their slice_pic_order_cnt_lsb values.

Five lists of picture order count values are constructed to derive the RPS. These five lists are PocStCurrBefore, PocStCurrAfter, PocStFoll, PocLtCurr and PocLtFoll, with NumPocStCurrBefore, NumPocStCurrAfter, NumPocStFoll, NumPocLtCurr and NumPocLtFoll number of elements, respectively. The five lists and the five variables are derived as follows:

– If the current picture is an IDR picture, PocStCurrBefore, PocStCurrAfter, PocStFoll, PocLtCurr and PocLtFoll are all set to be empty, and NumPocStCurrBefore, NumPocStCurrAfter, NumPocStFoll, NumPocLtCurr and NumPocLtFoll are all set equal to 0.

– Otherwise, the following applies:

```
for( i = 0, j = 0, k = 0; i < NumNegativePics[ CurrRpsIdx ] ; i++ )
    if( UsedByCurrPicS0[ CurrRpsIdx ][ i ] )
        PocStCurrBefore[ j++ ] = PicOrderCntVal + DeltaPocS0[ CurrRpsIdx ][ i ]
    else
        PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS0[ CurrRpsIdx ][ i ]
NumPocStCurrBefore = j
```

```
for( i = 0, j = 0; i < NumPositivePics[ CurrRpsIdx ]; i++ )
    if( UsedByCurrPicS1[ CurrRpsIdx ][ i ] )
        PocStCurrAfter[ j++ ] = PicOrderCntVal + DeltaPocS1[ CurrRpsIdx ][ i ]
    else
        PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS1[ CurrRpsIdx ][ i ]
NumPocStCurrAfter = j
NumPocStFoll = k                                                                    (8-5)
for( i = 0, j = 0, k = 0; i < num_long_term_sps + num_long_term_pics; i++ ) {
    pocLt = PocLsbLt[ i ]
    if( delta_poc_msb_present_flag[ i ] )
        pocLt += PicOrderCntVal − DeltaPocMsbCycleLt[ i ] * MaxPicOrderCntLsb −
                          ( PicOrderCntVal & ( MaxPicOrderCntLsb − 1 ) )
    if( UsedByCurrPicLt[ i ] ) {
        PocLtCurr[ j ] = pocLt
        CurrDeltaPocMsbPresentFlag[ j++ ] = delta_poc_msb_present_flag[ i ]
    } else {
        PocLtFoll[ k ] = pocLt
        FollDeltaPocMsbPresentFlag[ k++ ] = delta_poc_msb_present_flag[ i ]
    }
}
NumPocLtCurr = j
NumPocLtFoll = k
```

where PicOrderCntVal is the picture order count of the current picture as specified in clause 8.3.1.

NOTE 2 – A value of CurrRpsIdx in the range of 0 to num_short_term_ref_pic_sets − 1, inclusive, indicates that a candidate short-term RPS from the active SPS for the current layer is being used, where CurrRpsIdx is the index of the candidate short-term RPS into the list of candidate short-term RPSs signalled in the active SPS for the current layer. CurrRpsIdx equal to num_short_term_ref_pic_sets indicates that the short-term RPS of the current picture is directly signalled in the slice header.

For each i in the range of 0 to NumPocLtCurr − 1, inclusive, when CurrDeltaPocMsbPresentFlag[i] is equal to 1, it is a requirement of bitstream conformance that the following conditions apply:

– There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtCurr[i] is equal to PocStCurrBefore[j].

– There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtCurr[i] is equal to PocStCurrAfter[j].

– There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtCurr[i] is equal to PocStFoll[j].

– There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, where j is not equal to i, for which PocLtCurr[i] is equal to PocLtCurr[j].

For each i in the range of 0 to NumPocLtFoll − 1, inclusive, when FollDeltaPocMsbPresentFlag[i] is equal to 1, it is a requirement of bitstream conformance that the following conditions apply:

– There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtFoll[i] is equal to PocStCurrBefore[j].

– There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtFoll[i] is equal to PocStCurrAfter[j].

– There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtFoll[i] is equal to PocStFoll[j].

– There shall be no j in the range of 0 to NumPocLtFoll − 1, inclusive, where j is not equal to i, for which PocLtFoll[i] is equal to PocLtFoll[j].

– There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, for which PocLtFoll[i] is equal to PocLtCurr[j].

For each i in the range of 0 to NumPocLtCurr − 1, inclusive, when CurrDeltaPocMsbPresentFlag[i] is equal to 0, it is a requirement of bitstream conformance that the following conditions apply:

– There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtCurr[i] is equal to (PocStCurrBefore[j] & (MaxPicOrderCntLsb − 1)).

– There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtCurr[i] is equal to (PocStCurrAfter[j] & (MaxPicOrderCntLsb − 1)).

– There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtCurr[i] is equal to (PocStFoll[j] & (MaxPicOrderCntLsb − 1)).

– There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, where j is not equal to i, for which PocLtCurr[i] is equal to (PocLtCurr[j] & (MaxPicOrderCntLsb − 1)).

For each i in the range of 0 to NumPocLtFoll − 1, inclusive, when FollDeltaPocMsbPresentFlag[i] is equal to 0, it is a requirement of bitstream conformance that the following conditions apply:

– There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtFoll[i] is equal to (PocStCurrBefore[j] & (MaxPicOrderCntLsb − 1)).

– There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtFoll[i] is equal to (PocStCurrAfter[j] & (MaxPicOrderCntLsb − 1)).

– There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtFoll[i] is equal to (PocStFoll[j] & (MaxPicOrderCntLsb − 1)).

– There shall be no j in the range of 0 to NumPocLtFoll − 1, inclusive, where j is not equal to i, for which PocLtFoll[i] is equal to (PocLtFoll[j] & (MaxPicOrderCntLsb − 1)).

– There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, for which PocLtFoll[i] is equal to (PocLtCurr[j] & (MaxPicOrderCntLsb − 1)).

The variable NumPicTotalCurr is derived as specified in clause 7.4.7.2. It is a requirement of bitstream conformance that the following applies to the value of NumPicTotalCurr:

– If the current picture is a BLA or CRA picture, the value of NumPicTotalCurr shall be equal to pps_curr_pic_ref_enabled_flag.

– Otherwise, when the current picture contains a P or B slice, the value of NumPicTotalCurr shall not be equal to 0.

The RPS of the current picture consists of five RPS lists; RefPicSetStCurrBefore, RefPicSetStCurrAfter, RefPicSetStFoll, RefPicSetLtCurr and RefPicSetLtFoll. RefPicSetStCurrBefore, RefPicSetStCurrAfter and RefPicSetStFoll are collectively referred to as the short-term RPS. RefPicSetLtCurr and RefPicSetLtFoll are collectively referred to as the long-term RPS.

NOTE 3 – RefPicSetStCurrBefore, RefPicSetStCurrAfter and RefPicSetLtCurr contain all reference pictures that may be used for inter prediction of the current picture and one or more pictures that follow the current picture in decoding order. RefPicSetStFoll and RefPicSetLtFoll consist of all reference pictures that are *not* used for inter prediction of the current picture but may be used in inter prediction for one or more pictures that follow the current picture in decoding order.

The derivation process for the RPS and picture marking are performed according to the following ordered steps:

1. The following applies:

```
for( i = 0; i < NumPocLtCurr; i++ )
    if( !CurrDeltaPocMsbPresentFlag[ i ] )
        if( there is a reference picture picX in the DPB with
PicOrderCntVal & ( MaxPicOrderCntLsb − 1 )
                              equal to PocLtCurr[ i ] and nuh_layer_id equal to currPicLayerId )
            RefPicSetLtCurr[ i ] = picX
        else
            RefPicSetLtCurr[ i ] = "no reference picture"
    else
        if( there is a reference picture picX in the DPB with PicOrderCntVal equal to PocLtCurr[ i ]
                              and nuh_layer_id equal to currPicLayerId )
            RefPicSetLtCurr[ i ] = picX
        else
            RefPicSetLtCurr[ i ] = "no reference picture"
                                                                                    (8-6)
for( i = 0; i < NumPocLtFoll; i++ )
    if( !FollDeltaPocMsbPresentFlag[ i ] )
        if( there is a reference picture picX in the DPB with
```

```
PicOrderCntVal & ( MaxPicOrderCntLsb − 1 )
                              equal to PocLtFoll[ i ] and nuh_layer_id equal to currPicLayerId )
            RefPicSetLtFoll[ i ] = picX
        else
            RefPicSetLtFoll[ i ] = "no reference picture"
    else
        if( there is a reference picture picX in the DPB with PicOrderCntVal equal to PocLtFoll[ i ]
                              and nuh_layer_id equal to currPicLayerId )
            RefPicSetLtFoll[ i ] = picX
        else
            RefPicSetLtFoll[ i ] = "no reference picture"
```

2. All reference pictures that are included in RefPicSetLtCurr or RefPicSetLtFoll and have nuh_layer_id equal to currPicLayerId are marked as "used for long-term reference".

3. The following applies:

```
for( i = 0; i < NumPocStCurrBefore; i++ )
    if( there is a short-term reference picture picX in the DPB
            with PicOrderCntVal equal to PocStCurrBefore[ i ] and nuh_layer_id equal to
currPicLayerId )
        RefPicSetStCurrBefore[ i ] = picX
    else
        RefPicSetStCurrBefore[ i ] = "no reference picture"

for( i = 0; i < NumPocStCurrAfter; i++ )
    if( there is a short-term reference picture picX in the DPB
            with PicOrderCntVal equal to PocStCurrAfter[ i ] and nuh_layer_id equal to
currPicLayerId )
        RefPicSetStCurrAfter[ i ] = picX
    else
        RefPicSetStCurrAfter[ i ] = "no reference picture"                              (8-7)

for( i = 0; i < NumPocStFoll; i++ )
    if( there is a short-term reference picture picX in the DPB
            with PicOrderCntVal equal to PocStFoll[ i ] and nuh_layer_id equal to currPicLayerId )
        RefPicSetStFoll[ i ] = picX
    else
        RefPicSetStFoll[ i ] = "no reference picture"
```

4. All reference pictures in the DPB that are not included in RefPicSetLtCurr, RefPicSetLtFoll, RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetStFoll and have nuh_layer_id equal to currPicLayerId are marked as "unused for reference".

NOTE 4 – There may be one or more entries in the RPS lists that are equal to "no reference picture" because the corresponding pictures are not present in the DPB. Entries in RefPicSetStFoll or RefPicSetLtFoll that are equal to "no reference picture" should be ignored. An unintentional picture loss should be inferred for each entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr that is equal to "no reference picture".

NOTE 5 – A picture cannot be included in more than one of the five RPS lists.

It is a requirement of bitstream conformance that the RPS is restricted as follows:

– There shall be no entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr for which one or more of the following are true:
  – The entry is equal to "no reference picture".
  – The entry is an SLNR picture and has TemporalId equal to that of the current picture.
  – The entry is a picture that has TemporalId greater than that of the current picture.
– There shall be no entry in RefPicSetLtCurr or RefPicSetLtFoll for which the difference between the picture order count value of the current picture and the picture order count value of the entry is greater than or equal to 2^{24}.

– When the current picture is a temporal sub-layer access (TSA) picture, there shall be no picture included in the RPS with TemporalId greater than or equal to the TemporalId of the current picture.

– When the current picture is a step-wise temporal sub-layer access (STSA) picture, there shall be no picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that has TemporalId equal to that of the current picture.

– When the current picture is a picture that follows, in decoding order, an STSA picture that has TemporalId equal to that of the current picture, there shall be no picture that has TemporalId equal to that of the current picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that precedes the STSA picture in decoding order.

– When the current picture is a CRA picture, there shall be no picture included in the RPS that precedes, in output order or decoding order, any preceding IRAP picture in decoding order (when present).

– When the current picture is a trailing picture, there shall be no picture in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that was generated by the decoding process for generating unavailable reference pictures as specified in clause 8.3.3.

– When the current picture is a trailing picture, there shall be no picture in the RPS that precedes the associated IRAP picture in output order or decoding order.

– When the current picture is a RADL picture, there shall be no picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that is any of the following:

  – A RASL picture

  – A picture that was generated by the decoding process for generating unavailable reference pictures as specified in clause 8.3.3

  – A picture that precedes the associated IRAP picture in decoding order

– When sps_temporal_id_nesting_flag is equal to 1, the following applies:

  – Let tIdA be the value of TemporalId of the current picture picA.

  – Any picture picB with TemporalId equal to tIdB that is less than or equal to tIdA shall not be included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr of picA when there exists a picture picC that has TemporalId less than tIdB, follows picB in decoding order, and precedes picA in decoding order.

8.3.3 Decoding process for generating unavailable reference pictures

8.3.3.1 General decoding process for generating unavailable reference pictures

This process is invoked once per coded picture when the current picture is a BLA picture or is a CRA picture with NoRaslOutputFlag equal to 1.

NOTE – This process is primarily specified only for the specification of syntax constraints for RASL pictures. The entire specification of the decoding process for RASL pictures associated with an IRAP picture that has NoRaslOutputFlag equal to 1 is included herein only for purposes of specifying constraints on the allowed syntax content of such RASL pictures. During the decoding process, any RASL pictures associated with an IRAP picture that has NoRaslOutputFlag equal to 1 may be ignored, as these pictures are not specified for output and have no effect on the decoding process of any other pictures that are specified for output. However, in HRD operations as specified in Annex C, RASL access units may need to be taken into consideration in derivation of coded picture buffer (CPB) arrival and removal times.

When this process is invoked, the following applies:

– For each RefPicSetStFoll[i], with i in the range of 0 to NumPocStFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2, and the following applies:

  – The value of PicOrderCntVal for the generated picture is set equal to PocStFoll[i].

  – The value of PicOutputFlag for the generated picture is set equal to 0.

  – The generated picture is marked as "used for short-term reference".

  – RefPicSetStFoll[i] is set to be the generated reference picture.

  – The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id of the current picture.

– For each RefPicSetLtFoll[i], with i in the range of 0 to NumPocLtFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2, and the following applies:

  – The value of PicOrderCntVal for the generated picture is set equal to PocLtFoll[i].

- The value of slice_pic_order_cnt_lsb for the generated picture is inferred to be equal to (PocLtFoll[i] & (MaxPicOrderCntLsb − 1)).
- The value of PicOutputFlag for the generated picture is set equal to 0.
- The generated picture is marked as "used for long-term reference".
- RefPicSetLtFoll[i] is set to be the generated reference picture.
- The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id of the current picture.

8.3.3.2 Generation of one unavailable picture

When this process is invoked, an unavailable picture is generated as follows:

- The value of each element in the sample array S_L for the picture is set equal to 1 << ($BitDepth_Y$ − 1).
- When ChromaArrayType is not equal to 0, the value of each element in the sample arrays S_{Cb} and S_{Cr} for the picture is set equal to 1 << ($BitDepth_C$ − 1).
- The prediction mode CuPredMode[x][y] is set equal to MODE_INTRA for x = 0..pic_width_in_luma_samples − 1, y = 0..pic_height_in_luma_samples − 1.

8.3.4 Decoding process for reference picture lists construction

This process is invoked at the beginning of the decoding process for each P or B slice.

Reference pictures are addressed through reference indices as specified in clause 8.5.3.3.2. A reference index is an index into a reference picture list. When decoding a P slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.

At the beginning of the decoding process for each slice, the reference picture lists RefPicList0 and, for B slices, RefPicList1 are derived.

The variable NumRpsCurrTempList0 is set equal to Max(num_ref_idx_l0_active_minus1 + 1, NumPicTotalCurr) and the list RefPicListTemp0 is constructed as follows:

If TwoVersionsOfCurrDecPicFlag is equal to 1, let the variable currPic be the current decoded picture before the invocation of the in-loop filter process; otherwise (TwoVersionsOfCurrDecPicFlag is equal to 0), let the variable currPic be the current decoded picture after the invocation of the in-loop filter process. The variable NumRpsCurrTempList0 is set equal to Max(num_ref_idx_l0_active_minus1 + 1, NumPicTotalCurr) and the list RefPicListTemp0 is constructed as follows:

```
rIdx = 0
while( rIdx < NumRpsCurrTempList0 ) {
   for( i = 0; i < NumPocStCurrBefore  &&  rIdx < NumRpsCurrTempList0; rIdx++, i++ )
      RefPicListTemp0[ rIdx ] = RefPicSetStCurrBefore[ i ]
   for( i = 0;  i < NumPocStCurrAfter  &&  rIdx < NumRpsCurrTempList0; rIdx++, i++ )          (8-8)
      RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]
   for( i = 0; i < NumPocLtCurr  &&  rIdx < NumRpsCurrTempList0; rIdx++, i++ )
      RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]
   if( pps_curr_pic_ref_enabled_flag )
      RefPicListTemp0[ rIdx++ ] = currPic
}
```

The list RefPicList0 is constructed as follows:

```
for( rIdx = 0; rIdx  <=  num_ref_idx_l0_active_minus1; rIdx++)
   RefPicList0[ rIdx ] = ref_pic_list_modification_flag_l0 ? RefPicListTemp0[ list_entry_l0[ rIdx ] ] :
                                                              RefPicListTemp0[ rIdx ]
if( pps_curr_pic_ref_enabled_flag  &&  !ref_pic_list_modification_flag_l0  &&                  (8-9)
                           NumRpsCurrTempList0 > ( num_ref_idx_l0_active_minus1 + 1 ) )
   RefPicList0[ num_ref_idx_l0_active_minus1 ] = currPic
```

When the slice is a B slice, the variable NumRpsCurrTempList1 is set equal to Max(num_ref_idx_l1_active_minus1 + 1, NumPicTotalCurr) and the list RefPicListTemp1 is constructed as follows:

```
rIdx = 0
while( rIdx < NumRpsCurrTempList1 ) {
    for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrAfter[ i ]
    for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ )        (8-10)
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]
    for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]
    if( pps_curr_pic_ref_enabled_flag )
        RefPicListTemp1[ rIdx++ ] = currPic
}
```

When the slice is a B slice, the list RefPicList1 is constructed as follows:

```
for( rIdx = 0; rIdx <= num_ref_idx_l1_active_minus1; rIdx++)
    (8-11)
    RefPicList1[ rIdx ] = ref_pic_list_modification_flag_l1 ? RefPicListTemp1[ list_entry_l1[ rIdx ] ] :
                                                                RefPicListTemp1[ rIdx ]
```

It is a requirement of bitstream conformance that when nuh_layer_id is equal to 0, nal_unit_type has a value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive (i.e. the picture is an IRAP picture), pps_curr_pic_ref_enabled_flag is equal to 1, and slice_type is not equal to 2, RefPicList0 and RefPicList1 shall not contain entries that refer to a picture other than the current picture.

8.3.5 Decoding process for collocated picture and no backward prediction flag

This process is invoked at the beginning of the decoding process for each P or B slice, after decoding of the slice header as well as the invocation of the decoding process for reference picture set as specified in clause 8.3.2 and the invocation of the decoding process for reference picture list construction for the slice as specified in clause 8.3.4, but prior to the decoding of any coding unit.

When slice_temporal_mvp_enabled_flag is equal to 1, the variable ColPic is derived as follows:

– If slice_type is equal to B and collocated_from_l0_flag is equal to 0, ColPic is set equal to RefPicList1[collocated_ref_idx].

– Otherwise (slice_type is equal to B and collocated_from_l0_flag is equal to 1, or slice_type is equal to P), ColPic is set equal to RefPicList0[collocated_ref_idx].

The variable NoBackwardPredFlag is derived as follows:

– If DiffPicOrderCnt(aPic, CurrPic) is less than or equal to 0 for each picture aPic in RefPicList0 or RefPicList1 of the current slice, NoBackwardPredFlag is set equal to 1.

– Otherwise, NoBackwardPredFlag is set equal to 0.

8.4 Decoding process for coding units coded in intra prediction mode

8.4.1 General decoding process for coding units coded in intra prediction mode

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the luma location (xCb, yCb) as input.

A variable nCbS is set equal to 1 << log2CbSize.

When residual_adaptive_colour_transform_enabled_flag is equal to 1, the residual sample arrays resSamplesL, resSamplesCb, and resSamplesCr store the residual samples of the current coding unit.

Depending on the values of pcm_flag[xCb][yCb], palette_mode_flag[xCb][yCb], and IntraSplitFlag, the decoding process for luma samples is specified as follows:

– If pcm_flag[xCb][yCb] is equal to 1, the reconstructed picture is modified as follows:

$$S_L[\ xCb + i\ ][\ yCb + j\ ] = \text{pcm\_sample\_luma}[\ (\ nCbS * j\ ) + i\ ] \ll (\ BitDepth_Y - PcmBitDepth_Y\ ),\ \text{with } i, j = 0..nCbS - 1 \quad (8\text{-}12)$$

– Otherwise (pcm_flag[xCb][yCb] is equal to 0), if palette_mode_flag[xCb][yCb] is equal to 1, the following ordered steps apply:

1. The decoding process for palette intra blocks as specified in clause 8.4.4.2.7 is invoked with the luma location (xCb, yCb), the variable cIdx set equal to 0, and nCbSX and nCbSY both set equal to nCbS as inputs, and the output is an nCbS x nCbS array of reconstructed palette sample values, recSamples[x][y], x, y = 0..nCbS − 1.

2. The reconstructed picture is modified as follows for x, y = 0..nCbS − 1 and y = 0..nCbS − 1:

   – If palette_transpose_flag is equal to 1, the following applies:

$$SL[\ xCb + x\ ][\ yCb + y\ ] = recSamples[\ y\ ][\ x\ ] \quad (8\text{-}13)$$

   – Otherwise (palette_transpose_flag is equal to 0), the following applies:

$$SL[\ xCb + x\ ][\ yCb + y\ ] = recSamples[\ x\ ][\ y\ ] \quad (8\text{-}14)$$

– Otherwise (pcm_flag[xCb][yCb] is equal to 0, palette_mode_flag[xCb][yCb] is equal to 0), if IntraSplitFlag is equal to 0, the following ordered steps apply:

1. The derivation process for the intra prediction mode as specified in clause 8.4.2 is invoked with the luma location (xCb, yCb) as input.

2. If residual_adaptive_colour_transform_enabled_flag is equal to 1, the following applies:

   – The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (xCb, yCb), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeY[xCb][yCb], the variable cIdx set equal to 0 and variable controlParaAct set equal to 1 as inputs, and the output is a modified residual sample array resSamplesL.

   – The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xCb, yCb), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 1 and variable controlParaAct set equal to 1 as inputs, and the output is a modified residual sample array resSamplesCb.

   – The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xCb, yCb), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 2 and variable controlParaAct set equal to 1 as inputs, and the output is a modified residual sample array resSamplesCr.

   – The residual modification process for blocks using adaptive colour transform as specified in clause 8.6.8 is invoked with location (xCb, yCb), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable resSampleArrayL set equal to resSamplesL, the variable resSampleArrayCb set equal to resSamplesCb and the variable resSampleArrayCr set equal to resSamplesCr as inputs, and the outputs are modified versions of resSampleL, resSampleCb and resSampleCr.

3. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (xCb, yCb), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeY[xCb][yCb], the variable cIdx set equal to 0, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesL when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm_flag[xCb][yCb] is equal to 0, palette_mode_flag[xCb][yCb] is equal to 0, and IntraSplitFlag is equal to 1), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. The variable xPb is set equal to xCb + (nCbS >> 1) * (blkIdx % 2).

2. The variable yPb is set equal to yCb + (nCbS >> 1) * (blkIdx / 2).

3. The derivation process for the intra prediction mode as specified in clause 8.4.2 is invoked with the luma location (xPb, yPb) as input.

4. If residual_adaptive_colour_transform_enabled_flag is equal to 1, the following applies:

   – The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeY[xPb][yPb], the variable cIdx set equal to 0 and variable controlParaAct equal to 1 as inputs, and the output is a modified residual sample array resSamplesL.

   – The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 1 and variable controlParaAct equal to 1 as inputs, and the output is a modified residual sample array resSamplesCb.

   – The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 2 and variable controlParaAct set equal to 1 as inputs, and the output is a modified residual sample array resSamplesCr.

   – The residual modification process for blocks using adaptive colour transform as specified in clause 8.6.8 is invoked with location (xCb, yCb), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable resSampleArrayL set equal to resSamplesL, the variable resSampleArrayCb set equal to resSamplesCb and the variable resSampleArrayCr set equal to resSamplesCr as inputs, and the outputs are modified versions of resSampleL, resSampleCb and resSampleCr.

5. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeY[xPb][yPb], the variable cIdx set equal to 0, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesL when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

When ChromaArrayType is not equal to 0, the following applies.

The variable log2CbSizeC is set equal to log2CbSize − (ChromaArrayType = = 3 ? 0 : 1).

Depending on the values of pcm_flag[xCb][yCb] and IntraSplitFlag, the decoding process for chroma samples is specified as follows:

– If pcm_flag[xCb][yCb] is equal to 1, the reconstructed picture is modified as follows:

$$\begin{aligned}&S_{Cb}[\ xCb / SubWidthC + i\ ][\ yCb / SubHeightC + j\ ] =\\&\quad pcm\_sample\_chroma[\ (\ nCbS / SubWidthC * j\ ) + i\ ] \ll (\ BitDepth_C - PcmBitDepth_C\ ),\\&\quad \text{with } i = 0..nCbS / SubWidthC - 1 \text{ and } j = 0..nCbS / SubHeightC - 1\end{aligned}\tag{8-15}$$

$$\begin{aligned}&S_{Cr}[\ xCb / SubWidthC + i\ ][\ yCb / SubHeightC + j\ ] =\\&\quad pcm\_sample\_chroma[\ (\ nCbS / SubWidthC * (\ j + nCbS / SubHeightC\ )\ ) + i\ ] \ll\\&\quad (\ BitDepth_C - PcmBitDepth_C\ ),\\&\quad \text{with } i = 0..nCbS / SubWidthC - 1 \text{ and } j = 0..nCbS / SubHeightC - 1\end{aligned}\tag{8-16}$$

– Otherwise (pcm_flag[xCb][yCb] is equal to 0), if palette_mode_flag[xCb][yCb] is equal to 1 the following ordered steps apply:

1. nCbsX is set as follows:

   – If palette_transpose_flag is equal to 1, nCbSX is set equal to (nCbS / SubHeightC), nCbY is set equal to (nCbS / SubWidthC).

   – Otherwise (palette_transpose_flag is equal to 0) nCbSX is set equal to (nCbS / SubWidthC), nCbY is set equal to (nCbS / SubHeightC).

2. The decoding process for palette intra blocks as specified in clause 8.4.4.2.7 is invoked with the chroma location (xCb, yCb), the variable cIdx set equal to 1, nCbSX, and nCbSY as inputs, and the output is an (nCbS / SubWidthC) x (nCbS / SubHeightC) array of reconstructed palette sample values, recSamples[x][y], x = 0 … nCbS / SubWidthC − 1, y = 0..nCbS / SubHeightC − 1, when palette_transpose_flag is equal to 0, or an

(nCbS / SubHeightC) x (nCbS / SubWidthC) array of reconstructed palette sample values, recSamples[x][y], x = 0 … nCbS / SubHeightC − 1, y = 0..nCbS / SubWidthC − 1 when palette_transpose_flag is equal to 1.

3. The reconstructed picture is modified as follows:

– If palette_transpose_flag is equal to 1, the following applies:

SCb[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[y][x],
with x = 0..nCbS / SubWidthC − 1 and y = 0..nCbS / SubHeightC − 1 (8-17)

– Otherwise (palette_transpose_flag is equal to 0), the following applies:

SCb[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[x][y],
with x = 0..nCbS / SubWidthC − 1 and y = 0..nCbS / SubHeightC − 1 (8-18)

4. The decoding process for palette intra blocks as specified in clause 8.4.4.2.7 is invoked with the chroma location (xCb, yCb), the variable cIdx set equal to 2, nCbSX set equal to nCbS / SubWidthC, and nCbSY set equal to nCbS / SubHeightC as inputs, and the output is an (nCbS / SubWidthC) x (nCbS / SubHeightC) array of reconstructed palette sample values, recSamples[x][y], x = 0..nCbS / SubWidthC − 1, y = 0..nCbS / SubHeightC − 1, when palette_transpose_flag is equal to 0, or an (nCbS / SubHeightC) x (nCbS / SubWidthC) array of reconstructed palette sample values, recSamples[x][y], x = 0..nCbS / SubHeightC − 1, y = 0..nCbS / SubWidthC − 1, when palette_transpose_flag is equal to 1.

5. The reconstructed picture is modified as follows:

– If palette_transpose_flag is equal to 1, the following applies:

SCr[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[y][x],
with x = 0..nCbS / SubWidthC − 1 and y = 0..nCbS / SubHeightC − 1 (8-19)

– Otherwise (palette_transpose_flag is equal to 0), the following applies:

SCr[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[x][y],

with x = 0..nCbS / SubWidthC − 1 and y = 0..nCbS / SubHeightC − 1 (8-20)

– Otherwise (pcm_flag[xCb][yCb] is equal to 0 and palette_mode_flag[xCb][yCb] is equal to 0), if IntraSplitFlag is equal to 0 or ChromaArrayType is not equal to 3, the following ordered steps apply:

1. The derivation process for the chroma intra prediction mode as specified in clause 8.4.3 is invoked with the luma location (xCb, yCb) as input, and the output is the variable IntraPredModeC.

2. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xCb / SubWidthC, yCb / SubHeightC), the variable log2TrafoSize set equal to log2CbSizeC, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 1, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesCb when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

3. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xCb / SubWidthC, yCb / SubHeightC), the variable log2TrafoSize set equal to log2CbSizeC, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 2, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesCr when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm_flag[xCb][yCb] is equal to 0, palette_mode_flag[xCb][yCb] is equal to 0, IntraSplitFlag is equal to 1 and ChromaArrayType is equal to 3), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. The variable xPb is set equal to xCb + (nCbS >> 1) * (blkIdx % 2).

2. The variable yPb is set equal to yCb + (nCbS >> 1) * (blkIdx / 2).

3. The derivation process for the chroma intra prediction mode as specified in clause 8.4.3 is invoked with the luma location (xPb, yPb) as input, and the output is the variable IntraPredModeC.

4. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSizeC − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 1, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesCb when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

5. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSizeC − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 2, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesCr when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

8.4.2 Derivation process for luma intra prediction mode

Input to this process is a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

In this process, the luma intra prediction mode IntraPredModeY[xPb][yPb] is derived.

Table 8-1 specifies the value for the intra prediction mode and the associated names.

Table 8-1 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated name
0	INTRA_PLANAR
1	INTRA_DC
2..34	INTRA_ANGULAR2..INTRA_ANGULAR34

IntraPredModeY[xPb][yPb] labelled 0..34 represents directions of predictions as illustrated in Figure 8-1.

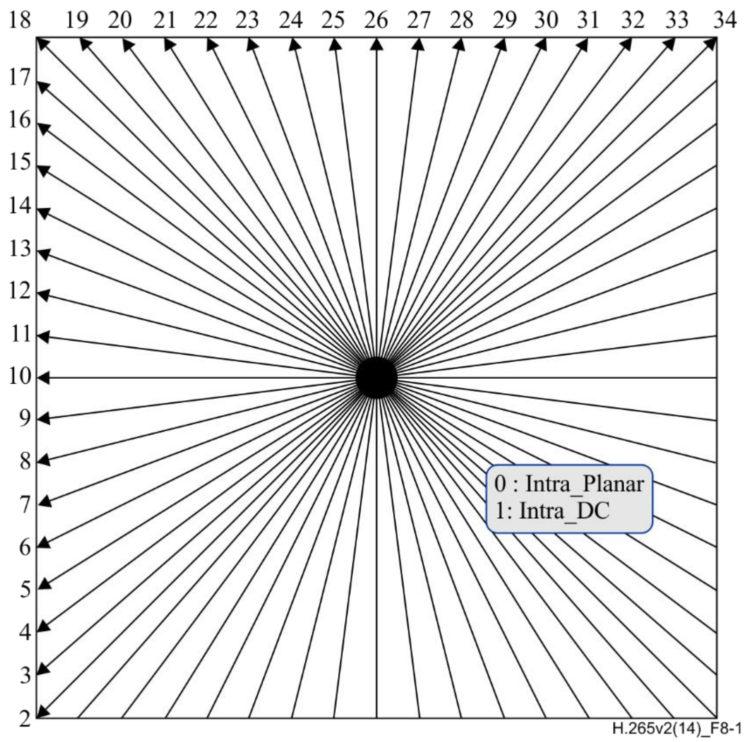


Figure 8-1 – Intra prediction mode directions (informative)

IntraPredModeY[xPb][yPb] is derived by the following ordered steps:

1. The neighbouring locations (xNbA, yNbA) and (xNbB, yNbB) are set equal to (xPb − 1, yPb) and (xPb, yPb − 1), respectively.
2. For X being replaced by either A or B, the variables candIntraPredModeX are derived as follows:
   – The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (xPb, yPb) and the neighbouring location (xNbY, yNbY) set equal to (xNbX, yNbX) as inputs, and the output is assigned to availableX.
   – The candidate intra prediction mode candIntraPredModeX is derived as follows:
     – If availableX is equal to FALSE, candIntraPredModeX is set equal to INTRA_DC.
     – Otherwise, if CuPredMode[xNbX][yNbX] is not equal to MODE_INTRA or pcm_flag[xNbX][yNbX] is equal to 1, candIntraPredModeX is set equal to INTRA_DC,
     – Otherwise, if X is equal to B and yPb − 1 is less than ((yPb >> CtbLog2SizeY) << CtbLog2SizeY), candIntraPredModeB is set equal to INTRA_DC.
     – Otherwise, candIntraPredModeX is set equal to IntraPredModeY[xNbX][yNbX].
3. The candModeList[x] with x = 0..2 is derived as follows:
   – If candIntraPredModeB is equal to candIntraPredModeA, the following applies:
     – If candIntraPredModeA is less than 2 (i.e., equal to INTRA_PLANAR or INTRA_DC), candModeList[x] with x = 0..2 is derived as follows:

$$\text{candModeList[0] = INTRA\_PLANAR} \tag{8-21}$$

$$\text{candModeList[1] = INTRA\_DC} \tag{8-22}$$

$$\text{candModeList[2] = INTRA\_ANGULAR26} \tag{8-23}$$

     – Otherwise, candModeList[x] with x = 0..2 is derived as follows:

$$\text{candModeList[0] = candIntraPredModeA} \tag{8-24}$$

$$\text{candModeList[1]} = 2 + ((\text{candIntraPredModeA} + 29) \% 32) \tag{8-25}$$

$$\text{candModeList[2]} = 2 + ((\text{candIntraPredModeA} - 2 + 1) \% 32) \tag{8-26}$$

   – Otherwise (candIntraPredModeB is not equal to candIntraPredModeA), the following applies:
     – candModeList[0] and candModeList[1] are derived as follows:

$$\text{candModeList[0] = candIntraPredModeA} \tag{8-27}$$

$$\text{candModeList[1] = candIntraPredModeB} \tag{8-28}$$

     – If neither of candModeList[0] and candModeList[1] is equal to INTRA_PLANAR, candModeList[2] is set equal to INTRA_PLANAR,
     – Otherwise, if neither of candModeList[0] and candModeList[1] is equal to INTRA_DC, candModeList[2] is set equal to INTRA_DC,
     – Otherwise, candModeList[2] is set equal to INTRA_ANGULAR26.
4. IntraPredModeY[xPb][yPb] is derived by applying the following procedure:
   – If prev_intra_luma_pred_flag[xPb][yPb] is equal to 1, the IntraPredModeY[xPb][yPb] is set equal to candModeList[mpm_idx[xPb][yPb]].
   – Otherwise, IntraPredModeY[xPb][yPb] is derived by applying the following ordered steps:
     1) The array candModeList[x], x = 0..2 is modified as the following ordered steps:
        i. When candModeList[0] is greater than candModeList[1], both values are swapped as follows:

(candModeList[0], candModeList[1]) = Swap(candModeList[0], candModeList[1]) (8-29)

ii. When candModeList[0] is greater than candModeList[2], both values are swapped as follows:

(candModeList[0], candModeList[2]) = Swap(candModeList[0], candModeList[2]) (8-30)

iii. When candModeList[1] is greater than candModeList[2], both values are swapped as follows:

(candModeList[1], candModeList[2]) = Swap(candModeList[1], candModeList[2]) (8-31)

2) IntraPredModeY[xPb][yPb] is derived by the following ordered steps:

i. IntraPredModeY[xPb][yPb] is set equal to rem_intra_luma_pred_mode[xPb][yPb].

ii. For i equal to 0 to 2, inclusive, when IntraPredModeY[xPb][yPb] is greater than or equal to candModeList[i], the value of IntraPredModeY[xPb][yPb] is incremented by one.

8.4.3 Derivation process for chroma intra prediction mode

This process is only invoked when ChromaArrayType is not equal to 0.

Input to this process is a luma location (xPb, yPb) specifying the top-left sample of the current chroma prediction block relative to the top-left luma sample of the current picture.

Output of this process is the variable IntraPredModeC.

The variable modeIdx is derived using intra_chroma_pred_mode[xPb][yPb] and IntraPredModeY[xPb][yPb] as specified in Table 8-2.

The chroma intra prediction mode IntraPredModeC is derived as follows:

– If ChromaArrayType is equal to 2, IntraPredModeC is set using modeIdx as specified in Table 8-3.

– Otherwise, IntraPredModeC is set equal to modeIdx.

Table 8-2 – Specification of modeIdx

intra_chroma_pred_mode[xPb][yPb]	IntraPredModeY[xPb][yPb]				
	0	26	10	1	X (0 <= X <= 34)
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

Table 8-3 – Specification of IntraPredModeC when ChromaArrayType is equal to 2

modeIdx	**X <= 2**		**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
IntraPredModeC	X		2	2	2	3	5	7	8	10	12	13	15	17	18	19	20
modeIdx	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
IntraPredModeC	21	22	23	23	24	24	25	25	26	27	27	28	28	29	29	30	31

8.4.4 Decoding process for intra blocks

8.4.4.1 General decoding process for intra blocks

Inputs to this process are:

- a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable log2TrafoSize specifying the size of the current transform block,
- a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,
- a variable predModeIntra specifying the intra prediction mode,
- a variable cIdx specifying the colour component of the current block,
- a variable controlParaAct specifying the applicable processes,
- when controlParaAct is equal to 2, a residual sample array resSamplesRec specifying the reconstructed residual samples for the current colour component of the current coding block.

Output of this process is a modified reconstructed picture before deblocking filtering when controlParaAct is not equal to 1, or a modified residual sample array resSampleArray for the current colour component of the current coding block when controlParaAct is equal to 1.

The luma sample location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture is derived as follows:

$$(xTbY, yTbY) = (cIdx == 0) ? (xTb0, yTb0) : (xTb0 * SubWidthC, yTb0 * SubHeightC) \quad (8\text{-}32)$$

The variable splitFlag is derived as follows:

- If cIdx is equal to 0, splitFlag is set equal to split_transform_flag[xTbY][yTbY][trafoDepth].
- Otherwise, if all of the following conditions are true, splitFlag is set equal to 1.
  - cIdx is greater than 0
  - split_transform_flag[xTbY][yTbY][trafoDepth] is equal to 1
  - log2TrafoSize is greater than 2
- Otherwise, splitFlag is set equal to 0.

Depending on the value of splitFlag, the following applies:

- If splitFlag is equal to 1, the following ordered steps apply:
  1. The variables xTb1 and yTb1 are derived as follows:
     - If cIdx is equal to 0 or ChromaArrayType is not equal to 2, the following applies:
       - The variable xTb1 is set equal to xTb0 + (1 << (log2TrafoSize − 1)).
       - The variable yTb1 is set equal to yTb0 + (1 << (log2TrafoSize − 1)).
     - Otherwise (ChromaArrayType is equal to 2 and cIdx is greater than 0), the following applies:
       - The variable xTb1 is set equal to xTb0 + (1 << (log2TrafoSize − 1)).
       - The variable yTb1 is set equal to yTb0 + (2 << (log2TrafoSize − 1)).
  2. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb0, yTb0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, the variable cIdx, the variable controlParaAct, and the array resSamplesRec when controlParaAct is equal to 2 as inputs and the output is a modified reconstructed picture before deblocking filtering when controlParaAct is not equal to 1 or a modified residual sample array resSampleArray when controlParaAct is equal to 1.
  3. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb1, yTb0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, the variable cIdx, the variable controlParaAct, and the array resSamplesRec when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed

picture before deblocking filtering when controlParaAct is not equal to 1 or a modified residual sample array resSampleArray when controlParaAct is equal to 1.

4. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb0, yTb1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, the variable cIdx, the variable controlParaAct, and the array resSamplesRec when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering when controlParaAct is not equal to 1 or a modified residual sample array resSampleArray when controlParaAct is equal to 1.

5. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb1, yTb1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, the variable cIdx, the variable controlParaAct, and the array resSamplesRec when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering when controlParaAct is not equal to 1 or a modified residual sample array resSampleArray when controlParaAct is equal to 1.

– Otherwise (splitFlag is equal to 0), for the variable blkIdx proceeding over the values 0..(cIdx > 0 && ChromaArrayType = = 2 ? 1 : 0), the following ordered steps apply:

1. The variable nTbS is set equal to 1 << log2TrafoSize.

2. The variable yTbOffset is set equal to blkIdx * nTbS.

3. The variable yTbOffsetY is set equal to yTbOffset * SubHeightC.

4. When controlParaAct is not equal to 2, the variable residualDpcm is derived as follows:

   – If all of the following conditions are true, residualDpcm is set equal to 1.

     – implicit_rdpcm_enabled_flag is equal to 1.

     – either transform_skip_flag[xTbY][yTbY + yTbOffsetY][cIdx] is equal to 1, or cu_transquant_bypass_flag is equal to 1.

     – either predModeIntra is equal to 10, or predModeIntra is equal to 26.

   – Otherwise, residualDpcm is set equal to 0.

5. When controlParaAct is not equal to 1, the general intra sample prediction process as specified in clause 8.4.4.2.1 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the intra prediction mode predModeIntra, the transform block size nTbS and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

6. When controlParaAct is not equal to 2, the scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location (xTbY, yTbY + yTbOffsetY), the variable trafoDepth, the variable cIdx and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array resSamples.

7. When controlParaAct is not equal to 2 and residualDpcm is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable mDir set equal to predModeIntra / 26, the variable nTbS and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.

8. When controlParaAct is not equal to 2 and cross_component_prediction_enabled_flag is equal to 1, ChromaArrayType is equal to 3, and cIdx is not equal to 0, the residual modification process for transform blocks using cross-component prediction as specified in clause 8.6.6 is invoked with the current luma transform block location (xTbY, yTbY), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array r_Y set equal to the corresponding luma residual sample array resSamples of the current transform block and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.

9. When controlParaAct is not equal to 0, the following applies:

   – The variable nCbS specifying the size of the current coding block is derived by setting nCbS equal to 1 << (log2TrafoSize + trafDepth).

   – The sample location (xTbInCb, yTbInCb) specifying the top-left sample of the current transform block relative to the top-left sample of the current coding block is derived by setting xTbInCb equal to xTb0 & (nCbS − 1) and setting yTbInCb equal to yTb0 & (nCbS − 1).

10. When controlParaAct is equal to 2, the (nTbS)x(nTbS) array resSamples is derived by setting resSamples[x][y] equal to resSamplesRec[x + xTbInCb][y + yTbInCb], for x and y in the range of 0 to nTbS − 1, inclusive.

11. The following applies:

   – If controlParaAct is equal to 1, the residual array resSamplesArray is modified by setting resSamplesArray[x + xTbInCb][y + yTbInCb] equal to resSamples[x][y], for x and y in the range of 0 to nTbS − 1, inclusive.

   – Otherwise (controlParaAct is not equal to 1), the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the variables nCurrSw and nCurrSh both set equal to nTbS, the variable cIdx, the (nTbS)x(nTbS) array predSamples and the (nTbS)x(nTbS) array resSamples as inputs.

8.4.4.2 Intra sample prediction

8.4.4.2.1 General intra sample prediction

Inputs to this process are:

– a sample location (xTbCmp, yTbCmp) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,

– a variable predModeIntra specifying the intra prediction mode,

– a variable nTbS specifying the transform block size,

– a variable cIdx specifying the colour component of the current block.

Outputs of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The nTbS * 4 + 1 neighbouring samples p[x][y] that are constructed samples prior to the deblocking filter process, with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1, are derived as follows:

– The neighbouring location (xNbCmp, yNbCmp) is specified by:

(xNbCmp, yNbCmp) = (xTbCmp + x, yTbCmp + y) (8-33)

– The current luma location (xTbY, yTbY) and the neighbouring luma location (xNbY, yNbY) are derived as follows:

(xTbY, yTbY) =
(cIdx = = 0) ? (xTbCmp, yTbCmp) : (xTbCmp * SubWidthC, yTbCmp * SubHeightC)
(8-34)

(xNbY, yNbY) =
(cIdx = = 0) ? (xNbCmp, yNbCmp) : (xNbCmp * SubWidthC, yNbCmp * SubHeightC)
(8-35)

– The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the current luma location (xCurr, yCurr) set equal to (xTbY, yTbY) and the neighbouring luma location (xNbY, yNbY) as inputs, and the output is assigned to availableN.

– Each sample p[x][y] is derived as follows:

   – If one or more of the following conditions are true, the sample p[x][y] is marked as "not available for intra prediction":

      – The variable availableN is equal to FALSE.

      – CuPredMode[xNbY][yNbY] is not equal to MODE_INTRA and constrained_intra_pred_flag is equal to 1.

   – Otherwise, the sample p[x][y] is marked as "available for intra prediction" and the sample at the location (xNbCmp, yNbCmp) is assigned to p[x][y].

When at least one sample p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 is marked as "not available for intra prediction", the reference sample substitution process for intra sample prediction in clause 8.4.4.2.2 is invoked with the samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1, nTbS and cIdx as inputs, and the modified samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 as output.

Depending on the value of predModeIntra, the following ordered steps apply:

1. When intra_smoothing_disabled_flag is equal to 0 and either cIdx is equal to 0 or ChromaArrayType is equal to 3, the filtering process of neighbouring samples specified in clause 8.4.4.2.3 is invoked with the sample array p, the transform block size nTbS and the colour component index cIdx as inputs, and the output is reassigned to the sample array p.

2. The intra sample prediction process according to predModeIntra applies as follows:

   – If predModeIntra is equal to INTRA_PLANAR, the corresponding intra prediction mode specified in clause 8.4.4.2.4 is invoked with the sample array p and the transform block size nTbS as inputs, and the output is the predicted sample array predSamples.

   – Otherwise, if predModeIntra is equal to INTRA_DC, the corresponding intra prediction mode specified in clause 8.4.4.2.5 is invoked with the sample array p, the transform block size nTbS and the colour component index cIdx as inputs, and the output is the predicted sample array predSamples.

   – Otherwise (predModeIntra is in the range of INTRA_ANGULAR2..INTRA_ANGULAR34), the corresponding intra prediction mode specified in clause 8.4.4.2.6 is invoked with the intra prediction mode predModeIntra, the sample array p, the transform block size nTbS and the colour component index cIdx as inputs, and the output is the predicted sample array predSamples.

8.4.4.2.2 Reference sample substitution process for intra sample prediction

Inputs to this process are:

– reference samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 for intra sample prediction,

– a transform block size nTbS, and

– a variable cIdx specifying the colour component of the current block.

Outputs of this process are the modified reference samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 for intra sample prediction.

The variable bitDepth is derived as follows:

– If cIdx is equal to 0, bitDepth is set equal to BitDepth_Y.

– Otherwise, bitDepth is set equal to BitDepth_C.

The values of the samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 are modified as follows:

– If all samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 are marked as "not available for intra prediction", the value 1 << (bitDepth − 1) is substituted for the values of all samples p[x][y].

– Otherwise (at least one but not all samples p[x][y] are marked as "not available for intra prediction"), the following ordered steps are performed:

  1. When p[−1][nTbS * 2 − 1] is marked as "not available for intra prediction", search sequentially starting from x = −1, y = nTbS * 2 − 1 to x = −1, y = −1, then from x = 0, y = −1 to x = nTbS * 2 − 1, y = −1. Once a sample p[x][y] marked as "available for intra prediction" is found, the search is terminated and the value of p[x][y] is assigned to p[−1][nTbS * 2 − 1].

  2. Search sequentially starting from x = −1, y = nTbS * 2 − 2 to x = −1, y = −1, when p[x][y] is marked as "not available for intra prediction", the value of p[x][y + 1] is substituted for the value of p[x][y].

  3. For x = 0..nTbS * 2 − 1, y = −1, when p[x][y] is marked as "not available for intra prediction", the value of p[x − 1][y] is substituted for the value of p[x][y].

All samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 are marked as "available for intra prediction".

8.4.4.2.3 Filtering process of neighbouring samples

Inputs to this process are:

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a variable nTbS specifying the transform block size.

Outputs of this process are the filtered samples pF[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1.

The variable filterFlag is derived as follows:

– If one or more of the following conditions are true, filterFlag is set equal to 0:
  – predModeIntra is equal to INTRA_DC.
  – nTbS is equal to 4.
– Otherwise, the following applies:
  – The variable minDistVerHor is set equal to Min(Abs(predModeIntra − 26), Abs(predModeIntra − 10)).
  – The variable intraHorVerDistThres[nTbS] is specified in Table 8-4.
  – The variable filterFlag is derived as follows:
    – If minDistVerHor is greater than intraHorVerDistThres[nTbS], filterFlag is set equal to 1.
    – Otherwise, filterFlag is set equal to 0.

Table 8-4 – Specification of intraHorVerDistThres[nTbS] for various transform block sizes

	nTbS = 8	**nTbS = 16**	**nTbS = 32**
intraHorVerDistThres[nTbS]	7	1	0

When filterFlag is equal to 1, the following applies:

– The variable biIntFlag is derived as follows:
  – If all of the following conditions are true, biIntFlag is set equal to 1:
    – strong_intra_smoothing_enabled_flag is equal to 1.
    – cIdx is equal to 0.
    – nTbS is equal to 32.
    – Abs(p[−1][−1] + p[nTbS * 2 − 1][−1] − 2 * p[nTbS − 1][−1]) is less than 1 << (BitDepth_Y − 5).
    – Abs(p[−1][−1] + p[−1][nTbS * 2 − 1] − 2 * p[−1][nTbS − 1]) is less than 1 << (BitDepth_Y − 5).
  – Otherwise, biIntFlag is set equal to 0.
– The filtering is performed as follows:
  – If biIntFlag is equal to 1, the filtered sample values pF[x][y] with x = −1, y = −1..63 and x = 0..63, y = −1 are derived as follows:

$$pF[-1][-1] = p[-1][-1] \quad (8\text{-}36)$$

$$pF[-1][y] = ((63 - y) * p[-1][-1] + (y + 1) * p[-1][63] + 32) >> 6 \text{ for } y = 0..62 \quad (8\text{-}37)$$

$$pF[-1][63] = p[-1][63] \quad (8\text{-}38)$$

$$pF[x][-1] = ((63 - x) * p[-1][-1] + (x + 1) * p[63][-1] + 32) >> 6 \text{ for } x = 0..62 \quad (8\text{-}39)$$

$$pF[63][-1] = p[63][-1] \quad (8\text{-}40)$$

  – Otherwise (biIntFlag is equal to 0), the filtered sample values pF[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 are derived as follows:

$$pF[-1][-1] = (p[-1][0] + 2 * p[-1][-1] + p[0][-1] + 2) >> 2 \quad (8\text{-}41)$$

$$pF[-1][y] = (p[-1][y + 1] + 2 * p[-1][y] + p[-1][y - 1] + 2) >> 2 \text{ for } y = 0..nTbS * 2 - 2 \quad (8\text{-}42)$$

$$pF[-1][nTbS * 2 - 1] = p[-1][nTbS * 2 - 1] \quad (8\text{-}43)$$

$$pF[\ x\ ][\ -1\ ] = (\ p[\ x-1\ ][\ -1\ ] + 2 * p[\ x\ ][\ -1\ ] + p[\ x+1\ ][\ -1\ ] + 2\ ) >> 2 \text{ for } x = 0..nTbS * 2 - 2 \quad (8\text{-}44)$$

$$pF[\ nTbS * 2 - 1\ ][\ -1\ ] = p[\ nTbS * 2 - 1\ ][\ -1\ ] \quad (8\text{-}45)$$

8.4.4.2.4 Specification of intra prediction mode INTRA_PLANAR

Inputs to this process are:

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a variable nTbS specifying the transform block size.

Outputs of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The values of the prediction samples predSamples[x][y], with x, y = 0..nTbS − 1, are derived as follows:

$$\begin{aligned} predSamples[\ x\ ][\ y\ ] = (\ (\ & nTbS - 1 - x\ ) * p[\ -1\ ][\ y\ ] + (\ x + 1\ ) * p[\ nTbS\ ][\ -1\ ] + \\ & (\ nTbS - 1 - y\ ) * p[\ x\ ][\ -1\ ] + \\ & (\ y + 1\ ) * p[\ -1\ ][\ nTbS\ ] + nTbS\ ) >> (\ Log2(\ nTbS\ ) + 1\ ) \end{aligned} \quad (8\text{-}46)$$

8.4.4.2.5 Specification of intra prediction mode INTRA_DC

Inputs to this process are:

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a variable nTbS specifying the transform block size,

– a variable cIdx specifying the colour component of the current block.

Outputs of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The values of the prediction samples predSamples[x][y], with x, y = 0..nTbS − 1, are derived by the following ordered steps:

1. A variable dcVal is derived as follows:

$$dcVal = \left(\sum_{x'=0}^{nTbS-1} p[\,x'\,][-1] + \sum_{y'=0}^{nTbS-1} p[-1][\,y'\,] + nTbS \right) >> (\ k+1\ ) \quad (8\text{-}47)$$

where k = Log2(nTbS).

2. Depending on the value of the colour component index cIdx, the following applies:

– If cIdx is equal to 0, intra_boundary_filtering_disabled_flag is equal to 0, and nTbS is less than 32, the following applies:

$$predSamples[\ 0\ ][\ 0\ ] = (\ p[\ -1\ ][\ 0\ ] + 2 * dcVal + p[\ 0\ ][\ -1\ ] + 2\ ) >> 2 \quad (8\text{-}48)$$

$$predSamples[\ x\ ][\ 0\ ] = (\ p[\ x\ ][\ -1\ ] + 3 * dcVal + 2\ ) >> 2, \text{ with } x = 1..nTbS - 1 \quad (8\text{-}49)$$

$$predSamples[\ 0\ ][\ y\ ] = (\ p[\ -1\ ][\ y\ ] + 3 * dcVal + 2\ ) >> 2, \text{ with } y = 1..nTbS - 1 \quad (8\text{-}50)$$

$$predSamples[\ x\ ][\ y\ ] = dcVal, \text{ with } x, y = 1..nTbS - 1 \quad (8\text{-}51)$$

– Otherwise, the prediction samples predSamples[x][y] are derived as follows:

$$predSamples[\ x\ ][\ y\ ] = dcVal, \text{ with } x, y = 0..nTbS - 1 \quad (8\text{-}52)$$

8.4.4.2.6 Specification of intra prediction mode in the range of INTRA_ANGULAR2.. INTRA_ANGULAR34

Inputs to this process are:

– the intra prediction mode predModeIntra,

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a variable nTbS specifying the transform block size,

– a variable cIdx specifying the colour component of the current block.

Outputs of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

Figure 8-2 illustrates the total 33 intra angles and Table 8-5 specifies the mapping table between predModeIntra and the angle parameter intraPredAngle.

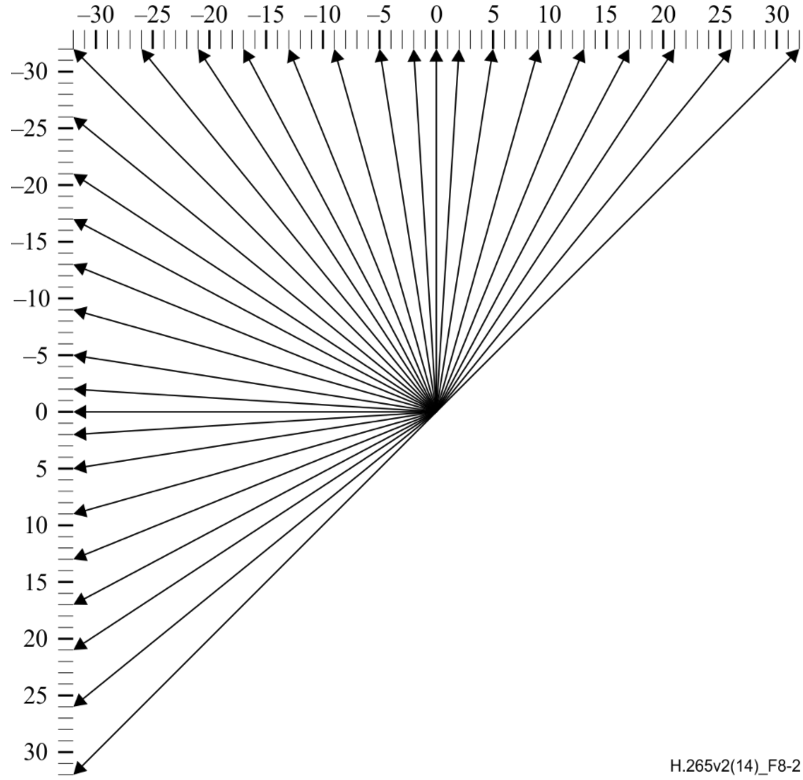


Figure 8-2 – Intra prediction angle definition (informative)

Table 8-5 – Specification of intraPredAngle

predModeIntra	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
intraPredAngle	-	32	26	21	17	13	9	5	2	0	−2	−5	−9	−13	−17	−21	−26
predModeIntra	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
intraPredAngle	−32	−26	−21	−17	−13	−9	−5	−2	0	2	5	9	13	17	21	26	32

Table 8-6 further specifies the mapping table between predModeIntra and the inverse angle parameter invAngle.

Table 8-6 – Specification of invAngle

predModeIntra	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
invAngle	−4 096	−1 638	−910	−630	−482	−390	−315	−256
predModeIntra	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**
invAngle	−315	−390	−482	−630	−910	−1 638	−4 096	-

The variable disableIntraBoundaryFilter is derived as follows:

– If intra_boundary_filtering_disabled_flag is equal to 1, disableIntraBoundaryFilter is set equal to 1.

– Otherwise (intra_boundary_filtering_disabled_flag is equal to 0), if implicit_rdpcm_enabled_flag and cu_transquant_bypass_flag are both equal to 1, disableIntraBoundaryFilter is set equal to 1.

– Otherwise, disableIntraBoundaryFilter is set equal to 0.

The values of the prediction samples predSamples[x][y], with x, y = 0..nTbS − 1 are derived as follows:

– If predModeIntra is greater than or equal to 18, the following ordered steps apply:

1. The reference sample array ref[x] is specified as follows:

   – The following applies:

   $$\text{ref}[\ x\ ] = \text{p}[\ -1 + x\ ][\ -1\ ],\ \text{with}\ x = 0..\text{nTbS} \tag{8-53}$$

   – If intraPredAngle is less than 0, the main reference sample array is extended as follows:

     – When (nTbS * intraPredAngle) >> 5 is less than −1,

     $$\text{ref}[\ x\ ] = \text{p}[\ -1\ ][\ -1 + (\ (\ x * \text{invAngle} + 128\ ) >> 8\ )\ ],\ \text{with}\ x = -1..(\ \text{nTbS} * \text{intraPredAngle}\ ) >> 5 \tag{8-54}$$

   – Otherwise,

   $$\text{ref}[\ x\ ] = \text{p}[\ -1 + x\ ][\ -1\ ],\ \text{with}\ x = \text{nTbS} + 1..2 * \text{nTbS} \tag{8-55}$$

2. The values of the prediction samples predSamples[x][y], with x, y = 0..nTbS − 1 are derived as follows:

   a. The index variable iIdx and the multiplication factor iFact are derived as follows:

   $$\text{iIdx} = (\ (\ y + 1\ ) * \text{intraPredAngle}\ ) >> 5 \tag{8-56}$$

   $$\text{iFact} = (\ (\ y + 1\ ) * \text{intraPredAngle}\ )\ \&\ 31 \tag{8-57}$$

   b. Depending on the value of iFact, the following applies:

     – If iFact is not equal to 0, the value of the prediction samples predSamples[x][y] is derived as follows:

     $$\text{predSamples}[\ x\ ][\ y\ ] = (\ (\ 32 - \text{iFact}\ ) * \text{ref}[\ x + \text{iIdx} + 1\ ] + \text{iFact} * \text{ref}[\ x + \text{iIdx} + 2\ ] + 16\ ) >> 5 \tag{8-58}$$

     – Otherwise, the value of the prediction samples predSamples[x][y] is derived as follows:

     $$\text{predSamples}[\ x\ ][\ y\ ] = \text{ref}[\ x + \text{iIdx} + 1\ ] \tag{8-59}$$

   c. When predModeIntra is equal to 26 (vertical), cIdx is equal to 0, nTbS is less than 32, and disableIntraBoundaryFilter is equal to 0, the following filtering applies with x = 0, y = 0..nTbS − 1:

   $$\text{predSamples}[\ x\ ][\ y\ ] = \text{Clip1}_Y(\ \text{p}[\ x\ ][\ -1\ ] + (\ (\ \text{p}[\ -1\ ][\ y\ ] - \text{p}[\ -1\ ][\ -1\ ]\ ) >> 1\ )\ ) \tag{8-60}$$

– Otherwise (predModeIntra is less than 18), the following ordered steps apply:

1. The reference sample array ref[x] is specified as follows:

   – The following applies:

   $$\text{ref}[\ x\ ] = \text{p}[\ -1\ ][\ -1 + x\ ],\ \text{with}\ x = 0..\text{nTbS} \tag{8-61}$$

   – If intraPredAngle is less than 0, the main reference sample array is extended as follows:

     – When (nTbS * intraPredAngle) >> 5 is less than −1,

$$\text{ref}[\,x\,] = \text{p}[\,-1 + (\,(\,x * \text{invAngle} + 128\,) \gg 8\,)\,][\,-1\,],\quad \text{with } x = -1..(\,\text{nTbS} * \text{intraPredAngle}\,) \gg 5 \tag{8-62}$$

– Otherwise,

$$\text{ref}[\,x\,] = \text{p}[\,-1\,][\,-1 + x\,],\ \text{with } x = \text{nTbS} + 1..2 * \text{nTbS} \tag{8-63}$$

2. The values of the prediction samples predSamples[x][y], with x, y = 0..nTbS − 1 are derived as follows:

a. The index variable iIdx and the multiplication factor iFact are derived as follows:

$$\text{iIdx} = (\,(\,x + 1\,) * \text{intraPredAngle}\,) \gg 5 \tag{8-64}$$

$$\text{iFact} = (\,(\,x + 1\,) * \text{intraPredAngle}\,)\ \&\ 31 \tag{8-65}$$

b. Depending on the value of iFact, the following applies:

– If iFact is not equal to 0, the value of the prediction samples predSamples[x][y] is derived as follows:

$$\text{predSamples}[\,x\,][\,y\,] = (\,(\,32 - \text{iFact}\,) * \text{ref}[\,y + \text{iIdx} + 1\,] + \text{iFact} * \text{ref}[\,y + \text{iIdx} + 2\,] + 16\,) \gg 5 \tag{8-66}$$

– Otherwise, the value of the prediction samples predSamples[x][y] is derived as follows:

$$\text{predSamples}[\,x\,][\,y\,] = \text{ref}[\,y + \text{iIdx} + 1\,] \tag{8-67}$$

c. When predModeIntra is equal to 10 (horizontal), cIdx is equal to 0, nTbS is less than 32 and disableIntraBoundaryFilter is equal to 0, the following filtering applies with x = 0..nTbS − 1, y = 0:

$$\text{predSamples}[\,x\,][\,y\,] = \text{Clip1}_{Y}(\,\text{p}[\,-1\,][\,y\,] + (\,(\,\text{p}[\,x\,][\,-1\,] - \text{p}[\,-1\,][\,-1\,]\,) \gg 1\,)\,) \tag{8-68}$$

8.4.4.2.7 Decoding process for palette mode

Inputs to this process are:

– a location (xCb, yCb) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current picture,

– a variable cIdx specifying the colour component of the current block,

– two variables nCbSX and nCbSY specifying the width and height of the current block, respectively.

Output of this process is an array recSamples[x][y], with x = 0..nCbSX − 1, y = 0..nCbSY − 1 specifying reconstructed sample values for the block.

Depending on the value of cIdx, the variables nSubWidth and nSubHeight are derived as follows:

– If cIdx is equal to 0, nSubWidth is set to 1 and nSubHeight is set to 1.

– Otherwise, nSubWidth is set to SubWidthC and nSubHeight is set to SubHeightC.

The (nCbSX x nCbSY) block of the reconstructed sample array recSamples at location (xCb, yCb) is represented by recSamples[x][y] with x = 0..nCbSX − 1 and y = 0..nCbSY − 1, and the value of recSamples[x][y] for each x in the range of 0 to nCbSX − 1, inclusive, and each y in the range of 0 to nCbSY − 1, inclusive, is derived as follows:

– The variables xL and yL are derived as follows:

$$\text{xL} = \text{palette\_transpose\_flag}\ ?\ y * \text{nSubHeight} : x * \text{nSubWidth} \tag{8-69}$$

$$\text{yL} = \text{palette\_transpose\_flag}\ ?\ x * \text{nSubWidth} : y * \text{nSubHeight} \tag{8-70}$$

– The variable bIsEscapeSample is derived as follows:

– If PaletteIndexMap[xCb + xL][yCb + yL] is equal to MaxPaletteIndex and palette_escape_val_present_flag is equal to 1, bIsEscapeSample is set equal to 1.

– Otherwise, bIsEscapeSample is set equal to 0.

– If bIsEscapeSample is equal to 0, the following applies:

recSamples[x][y] = CurrentPaletteEntries[cIdx][PaletteIndexMap[xCb + xL][yCb + yL]] (8-71)

– Otherwise, if cu_transquant_bypass_flag is equal to 1, the following applies:

recSamples[x][y] = PaletteEscapeVal[cIdx][xCb + xL][yCb + yL] (8-72)

– Otherwise (bIsEscapeSample is equal to 1 and cu_transquant_bypass_flag is equal to 0), the following ordered steps apply:

1. The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the location (xCb, yCb) specifying the top-left sample of the current block relative to the top-left sample of the current picture.

2. The quantization parameter qP is derived as follows:

   – If cIdx is equal to 0,

   qP = Max(0, Qp′Y) (8-73)

   – Otherwise, if cIdx is equal to 1,

   qP = Max(0, Qp′Cb) (8-74)

   – Otherwise (cIdx is equal to 2),

   qP = Max(0, Qp′Cr) (8-75)

3. The variable bitDepth is derived as follows:

   $\text{bitDepth} = (\text{cIdx} == 0)\ ?\ \text{BitDepth}_Y : \text{BitDepth}_C$ (8-76)

4. The list levelScale[ ] is specified as levelScale[k] = { 40, 45, 51, 57, 64, 72 } with k = 0..5.

5. The following applies:

   tmpVal = (PaletteEscapeVal[cIdx][xCb + xL][yCb + yL] *
   levelScale[qP%6]) << (qP / 6) + 32) >> 6 (8-77)

   recSamples[x][y] = Clip3(0, (1 << bitDepth) − 1, tmpVal) (8-78)

The variable PredictorPaletteSize and the array PredictorPaletteEntries are derived or modified as follows:

```
numComps = ( ChromaArrayType == 0 ) ? 1 : 3
for( i = 0; i < CurrentPaletteSize; i++ )
   for( cIdx = 0; cIdx < numComps; cIdx++ )
      newPredictorPaletteEntries[ cIdx ][ i ] = CurrentPaletteEntries[ cIdx ][ i ]
newPredictorPaletteSize = CurrentPaletteSize
for( i = 0; i < PredictorPaletteSize && newPredictorPaletteSize < PaletteMaxPredictorSize; i++ )
   if( !PalettePredictorEntryReuseFlags[ i ] ) {
      for( cIdx = 0; cIdx < numComps; cIdx++ )
   (8-79)
         newPredictorPaletteEntries[ cIdx ][ newPredictorPaletteSize ] =
            PredictorPaletteEntries[ cIdx ][ i ]
      newPredictorPaletteSize++
   }
for( cIdx = 0; cIdx < numComps; cIdx++ )
```

```
    for( i = 0; i < newPredictorPaletteSize; i++ )
        PredictorPaletteEntries[ cIdx ][ i ] = newPredictorPaletteEntries[ cIdx ][ i ]
PredictorPaletteSize = newPredictorPaletteSize
```

It is a requirement of bitstream conformance that the value of PredictorPaletteSize shall be in the range of 0 to PaletteMaxPredictorSize, inclusive.

8.5 Decoding process for coding units coded in inter prediction mode

8.5.1 General decoding process for coding units coded in inter prediction mode

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the luma location (xCb, yCb) as input.

The variable $nCbS_L$ is set equal to 1 << log2CbSize. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to (1 << log2CbSize) / SubWidthC and the variable $nCbSh_C$ is set equal to (1 << log2CbSize) / SubHeightC.

The decoding process for coding units coded in inter prediction mode consists of the following ordered steps:

1. The inter prediction process as specified in clause 8.5.2 is invoked with the luma location (xCb, yCb) and the luma coding block size log2CbSize as inputs, and the outputs are the array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for the residual signal of coding units coded in inter prediction mode specified in clause 8.5.4 is invoked with the luma location (xCb, yCb) and the luma coding block size log2CbSize as inputs, and the outputs are the array $resSamples_L$ and, when ChromaArrayType is not equal to 0, the arrays $resSamples_{Cb}$ and $resSamples_{Cr}$.

3. The reconstructed samples of the current coding unit are derived as follows:

   – The picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the luma coding block location (xCb, yCb), the variable nCurrSw set equal to $nCbS_L$, the variable nCurrSh set equal to $nCbS_L$, the variable cIdx set equal to 0, the $(nCbS_L)$x$(nCbS_L)$ array predSamples set equal to $predSamples_L$ and the $(nCbS_L)$x$(nCbS_L)$ array resSamples set equal to $resSamples_L$ as inputs.

   – When ChromaArrayType is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location (xCb / SubWidthC, yCb / SubHeightC), the variable nCurrSw set equal to $nCbSw_C$, the variable nCurrSh set equal to $nCbSh_C$, the variable cIdx set equal to 1, the $(nCbSw_C)$x$(nCbSh_C)$ array predSamples set equal to $predSamples_{Cb}$ and the $(nCbSw_C)$x$(nCbSh_C)$ array resSamples set equal to $resSamples_{Cb}$ as inputs.

   – When ChromaArrayType is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location (xCb / SubWidthC, yCb / SubHeightC), the variable nCurrSw set equal to $nCbSw_C$, the variable nCurrSh set equal to $nCbSh_C$, the variable cIdx set equal to 2, the $(nCbSw_C)$x$(nCbSh_C)$ array predSamples set equal to $predSamples_{Cr}$ and the $(nCbSw_C)$x$(nCbSh_C)$ array resSamples set equal to $resSamples_{Cr}$ as inputs.

8.5.2 Inter prediction process

This process is invoked when decoding coding unit whose CuPredMode[xCb][yCb] is not equal to MODE_INTRA.

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block.

Outputs of this process are:

– an $(nCbS_L)$x$(nCbS_L)$ array $predSamples_L$ of luma prediction samples, where $nCbS_L$ is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to 1 << log2CbSize. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS_L$ / SubWidthC and the variable $nCbSh_C$ is set equal to $nCbS_L$ / SubHeightC.

The variable $nCbS1_L$ is set equal to $nCbS_L$ >> 1.

Depending on the value of PartMode, the following applies:

– If PartMode is equal to PART_2Nx2N, the decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$, the height of the luma prediction block nPbH set equal to $nCbS_L$ and a partition index partIdx set equal to 0 as inputs, and the outputs are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise, if PartMode is equal to PART_2NxN, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 1 and a partition index partIdx set equal to 0 as inputs, and the outputs are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, $nCbS_L$ >> 1), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 1 and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise, if PartMode is equal to PART_Nx2N, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 1, the height of the luma prediction block nPbH set equal to $nCbS_L$ and a partition index partIdx set equal to 0 as inputs, and the outputs are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to ($nCbS_L$ >> 1, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 1, the height of the luma prediction block nPbH set equal to $nCbS_L$ and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise, if PartMode is equal to PART_2NxnU, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 2 and a partition index partIdx set equal to 0 as inputs, and the outputs are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, $nCbS_L$ >> 2), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$, the height of the luma prediction block nPbH set equal to ($nCbS_L$ >> 1) + ($nCbS_L$ >> 2) and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise, if PartMode is equal to PART_2NxnD, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$, the height of the luma prediction block nPbH set equal to ($nCbS_L$ >> 1) + ($nCbS_L$ >> 2) and a partition index partIdx set equal to 0 as inputs, and the outputs are an ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, ($nCbS_L$ >> 1) + ($nCbS_L$ >> 2)), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 2 and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise, if PartMode is equal to PART_nLx2N, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 2, the height of the luma prediction block nPbH set equal to $nCbS_L$ and a partition index partIdx set equal to 0 as inputs, and the outputs are an ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to ($nCbS_L$ >> 2, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to ($nCbS_L$ >> 1) + ($nCbS_L$ >> 2), the height of the luma prediction block nPbH set equal to $nCbS_L$ and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise, if PartMode is equal to PART_nRx2N, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to ($nCbS_L$ >> 1) + ($nCbS_L$ >> 2), the height of the luma prediction block nPbH set equal to $nCbS_L$ and a partition index partIdx set equal to 0 as inputs, and the outputs are an ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (($nCbS_L$ >> 1) + ($nCbS_L$ >> 2), 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 2, the height of the luma prediction block nPbH set equal to $nCbS_L$ and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise (PartMode is equal to PART_NxN), the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 1, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 1 and a partition index partIdx set equal to 0 as inputs, and the outputs are an ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to ($nCbS_L$ >> 1, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 1, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 1 and a partition index partIdx set equal to 1 as inputs, and the outputs are the modified ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

3. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to (0, $nCbS_L$ >> 1), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 1, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 1 and a partition index partIdx set equal to 2 as inputs, and the outputs are the modified ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

4. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBl, yBl) set equal to ($nCbS_L$ >> 1, $nCbS_L$ >> 1), the size of the luma coding block $nCbS_L$, the width of the luma prediction block nPbW set equal to $nCbS_L$ >> 1, the height of the luma prediction block nPbH set equal to $nCbS_L$ >> 1 and a partition index partIdx set equal to 3 as inputs, and the outputs are the modified ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

8.5.3 Decoding process for prediction units in inter prediction mode

8.5.3.1 General

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– a variable nPbW specifying the width of the current luma prediction block,

– a variable nPbH specifying the height of the current luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– an ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ of luma prediction samples, where $nCbS_L$ is derived as specified below,

– when ChromaArrayType is not equal to 0, an ($nCbSw_C$)x($nCbSh_C$) array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,

– when ChromaArrayType is not equal to 0, an ($nCbSw_C$)x($nCbSh_C$) array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to nCbS. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to nCbS / SubWidthC and the variable $nCbSh_C$ is set equal to nCbS / SubHeightC.

The decoding process for prediction units in inter prediction mode consists of the following ordered steps:

1. The derivation process for motion vector components and reference indices as specified in clause 8.5.3.2 is invoked with the luma coding block location (xCb, yCb), the luma prediction block location (xBl, yBl), the luma coding block size block nCbS, the luma prediction block width nPbW, the luma prediction block height nPbH and the prediction unit index partIdx as inputs, and the luma motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1 and the prediction list utilization flags predFlagL0 and predFlagL1 as outputs.

2. The decoding process for inter sample prediction as specified in clause 8.5.3.3 is invoked with the luma coding block location (xCb, yCb), the luma prediction block location (xBl, yBl), the luma coding block size block nCbS, the luma prediction block width nPbW, the luma prediction block height nPbH, the luma motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1 as inputs, and the inter prediction samples (predSamples) that are an ($nCbS_L$)x($nCbS_L$) array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two ($nCbSw_C$)x($nCbSh_C$) arrays $predSamples_{Cr}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr, as outputs.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xBl..xBl + nPbW − 1 and y = yBl..yBl + nPbH − 1:

$$\text{MvL0}[\ xCb + x\ ][\ yCb + y\ ] = \text{mvL0} \tag{8-80}$$

$$\text{MvL1}[\ xCb + x\ ][\ yCb + y\ ] = \text{mvL1} \tag{8-81}$$

$$\text{RefIdxL0}[\ xCb + x\ ][\ yCb + y\ ] = \text{refIdxL0} \tag{8-82}$$

$$\text{RefIdxL1}[\ xCb + x\ ][\ yCb + y\ ] = \text{refIdxL1} \tag{8-83}$$

$$\text{PredFlagL0[xCb + x][yCb + y] = predFlagL0} \quad (8\text{-}84)$$

$$\text{PredFlagL1[xCb + x][yCb + y] = predFlagL1} \quad (8\text{-}85)$$

8.5.3.2 Derivation process for motion vector components and reference indices

8.5.3.2.1 General

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) of the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– the luma motion vectors mvL0 and mvL1,

– when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1.

Let (xPb, yPb) specify the top-left sample location of the current luma prediction block relative to the top-left luma sample of the current picture where xPb = xCb + xBl and yPb = yCb + yBl.

Let the variable LX be RefPicListX, with X being 0 or 1, of the current picture.

The variables nPbSw, and nPbSh are derived as follows:

$$\text{nPbSw = nCbS / ((PartMode == PART\_2Nx2N || PartMode == PART\_2NxN) ? 1 : 2)} \quad (8\text{-}86)$$

$$\text{nPbSh = nCbS / ((PartMode == PART\_2Nx2N || PartMode == PART\_Nx2N) ? 1 : 2)} \quad (8\text{-}87)$$

The function LongTermRefPic(aPic, aPb, refIdx, LX), with X being 0 or 1, is defined as follows:

– If the picture with index refIdx from reference picture list LX of the slice containing prediction block aPb in the picture aPic was marked as "used for long-term reference" at the time when aPic was the current picture, LongTermRefPic(aPic, aPb, refIdx, LX) is equal to 1.

– Otherwise, LongTermRefPic(aPic, aPb, refIdx, LX) is equal to 0.

For the derivation of the variables mvL0 and mvL1, refIdxL0 and refIdxL1, as well as predFlagL0 and predFlagL1, the following applies:

– If merge_flag[xPb][yPb] is equal to 1, the derivation process for luma motion vectors for merge mode as specified in clause 8.5.3.2.2 is invoked with the luma location (xCb, yCb), the luma location (xPb, yPb), the variables nCbS, nPbW, nPbH and the partition index partIdx as inputs, and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1 and the prediction list utilization flags predFlagL0 and predFlagL1.

– Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX and refIdxLX, in PRED_LX, and in the syntax elements ref_idx_lX and MvdLX, the following ordered steps apply:

  1. The variables refIdxLX and predFlagLX are derived as follows:

     − If inter_pred_idc[xPb][yPb] is equal to PRED_LX or PRED_BI,

$$\text{refIdxLX = ref\_idx\_lX[xPb][yPb]} \quad (8\text{-}88)$$

$$\text{predFlagLX = 1} \quad (8\text{-}89)$$

     − Otherwise, the variables refIdxLX and predFlagLX are specified by:

$$\text{refIdxLX} = -1 \tag{8-90}$$

$$\text{predFlagLX} = 0 \tag{8-91}$$

2. The variable mvdLX is derived as follows:

$$\text{mvdLX[0]} = \text{MvdLX[xPb][yPb][0]} \tag{8-92}$$

$$\text{mvdLX[1]} = \text{MvdLX[xPb][yPb][1]} \tag{8-93}$$

3. When predFlagLX is equal to 1, the derivation process for luma motion vector prediction in clause 8.5.3.2.6 is invoked with the luma coding block location (xCb, yCb), the coding block size nCbS, the luma prediction block location (xPb, yPb), the variables nPbW, nPbH, refIdxLX and the partition index partIdx as inputs, and the output being mvpLX.

4. When predFlagLX is equal to 1 and the picture with index refIdx from reference picture list LX of the slice is not the current picture, and use_integer_mv_flag is equal to 0, the luma motion vector mvLX is derived as follows:

$$\text{uLX[0]} = (\text{ mvpLX[0]} + \text{mvdLX[0]} + 2^{16})\ \%\ 2^{16} \tag{8-94}$$

$$\text{mvLX[0]} = (\text{ uLX[0]} >= 2^{15})\ ?\ (\text{ uLX[0]} - 2^{16}) : \text{uLX[0]} \tag{8-95}$$

$$\text{uLX[1]} = (\text{ mvpLX[1]} + \text{mvdLX[1]} + 2^{16})\ \%\ 2^{16} \tag{8-96}$$

$$\text{mvLX[1]} = (\text{ uLX[1]} >= 2^{15})\ ?\ (\text{ uLX[1]} - 2^{16}) : \text{uLX[1]} \tag{8-97}$$

NOTE 1– The resulting values of mvLX[0] and mvLX[1] as specified above will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

5. When predFlagLX is equal to 1 and the reference picture is the current picture, or when predFlagLX is equal to 1 and use_integer_mv_flag is equal to 1, the luma motion vector mvLX is derived as follows:

$$\text{uLX[0]} = (\ (\ (\ (\text{ mvpLX[0]} >> 2) + \text{mvdLX[0]}) << 2) + 2^{16})\ \%\ 2^{16} \tag{8-98}$$

$$\text{mvLX[0]} = (\text{ uLX[0]} >= 2^{15})\ ?\ (\text{ uLX[0]} - 2^{16}) : \text{uLX[0]} \tag{8-99}$$

$$\text{uLX[1]} = (\ (\ (\ (\text{ mvpLX[1]} >> 2) + \text{mvdLX[1]}) << 2) + 2^{16})\ \%\ 2^{16} \tag{8-100}$$

$$\text{mvLX[1]} = (\text{ uLX[1]} >= 2^{15})\ ?\ (\text{ uLX[1]} - 2^{16}) : \text{uLX[1]} \tag{8-101}$$

NOTE 2 – The resulting values of mvLX[0] and mvLX[1] as specified above will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

– The variable noIntegerMvFlag is derived as follows:

$$\text{noIntegerMvFlag} = !(\ (\text{ mvL0 \& 0x3 } == 0)\ ||\ (\text{ mvL1 \& 0x3 } == 0)\ ) \tag{8-102}$$

– The variable identicalMvs is derived as follows:

$$\text{identicalMvs} = (\text{ mvL0 } == \text{ mvL1 })\ \&\&\ (\text{ DiffPicOrderCnt(RefPicList0[refIdxL0], RefPicList1[refIdxL1])} == 0) \tag{8-103}$$

When all of the following conditions are true, refIdxL1 is set equal to −1 and predFlagL1 is set equal to 0.

– predFlagL0 is equal to 1.
– predFlagL1 is equal to 1.
– nPbSw is equal to 8.
– nPbSh is equal to 8.
– TwoVersionsOfCurrDecPicFlag is equal to 1.

– noIntegerMvFlag is equal to 1.

– identicalMvs is equal to 0.

When ChromaArrayType is not equal to 0 and predFlagLX, with X being 0 or 1, is equal to 1, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with mvLX as input, and the output being mvCLX.

The variables offsetX and offsetY are derived as follows:

$$\text{offsetX} = (\ \text{ChromaArrayType}\ ==\ 0\ )\ ?\ 0 : (\ \text{mvCLX}[\ 0\ ]\ \&\ 0\text{x}7\ ?\ 2 : 0\ ) \qquad (8\text{-}104)$$

$$\text{offsetY} = (\ \text{ChromaArrayType}\ ==\ 0\ )\ ?\ 0 : (\ \text{mvCLX}[\ 1\ ]\ \&\ 0\text{x}7\ ?\ 2 : 0\ ) \qquad (8\text{-}105)$$

It is a requirement of bitstream conformance that when the reference picture is the current picture, the luma motion vector mvLX shall obey the following constraints:

– When the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring luma location (xNbY, yNbY) set equal to (xPb + (mvLX[0] >> 2) − offsetX, yPb + (mvLX[1] >> 2) − offsetY) as inputs, the output shall be equal to TRUE.

– When the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring luma location (xNbY, yNbY) set equal to (xPb + (mvLX[0] >> 2) + nPbW − 1 + offsetX, yPb + (mvLX[1] >> 2) + nPbH − 1 + offsetY) as inputs, the output shall be equal to TRUE.

– One or both of the following conditions shall be true:

  – The value of (mvLX[0] >> 2) + nPbW + xB1 + offsetX is less than or equal to 0.

  – The value of (mvLX[1] >> 2) + nPbH + yB1 + offsetY is less than or equal to 0.

– The following condition shall be true:

$$\begin{aligned}&(\ \text{xPb} + (\ \text{mvLX}[\ 0\ ]\ >>\ 2\ ) + \text{nPbSw} - 1 + \text{offsetX}\ ) / \text{CtbSizeY} - \text{xCb} / \text{CtbSizeY}\ <= \\ &\quad \text{yCb}/\text{CtbSizeY} - (\ \text{yPb} + (\ \text{mvLX}[\ 1\ ]\ >>\ 2\ ) + \text{nPbSh} - 1 + \text{offsetY}\ ) / \text{CtbSizeY}\end{aligned} \qquad (8\text{-}106)$$

8.5.3.2.2 Derivation process for luma motion vectors for merge mode

This process is only invoked when merge_flag[xPb][yPb] is equal to 1, where (xPb, yPb) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– the luma motion vectors mvL0 and mvL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1.

The location (xOrigP, yOrigP) and the variables nOrigPbW and nOrigPbH are derived to store the values of (xPb, yPb), nPbW and nPbH as follows:

(xOrigP, yOrigP) is set equal to (xPb, yPb) (8-107)

$$\text{nOrigPbW} = \text{nPbW} \qquad (8\text{-}108)$$

$$\text{nOrigPbH} = \text{nPbH} \tag{8-109}$$

When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, (xPb, yPb), nPbW, nPbH and partIdx are modified as follows:

$$(\text{ xPb, yPb }) = (\text{ xCb, yCb }) \tag{8-110}$$

$$\text{nPbW} = \text{nCbS} \tag{8-111}$$

$$\text{nPbH} = \text{nCbS} \tag{8-112}$$

$$\text{partIdx} = 0 \tag{8-113}$$

NOTE – When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, all the prediction units of the current coding unit share a single merge candidate list, which is identical to the merge candidate list of the 2Nx2N prediction unit.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1 and the prediction utilization flags predFlagL0 and predFlagL1 are derived by the following ordered steps:

1. The derivation process for merging candidates from neighbouring prediction unit partitions in clause 8.5.3.2.3 is invoked with the luma coding block location (xCb, yCb), the coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and the partition index partIdx as inputs, and the output being the availability flags availableFlagA_0, availableFlagA_1, availableFlagB_0, availableFlagB_1 and availableFlagB_2, the reference indices refIdxLXA_0, refIdxLXA_1, refIdxLXB_0, refIdxLXB_1 and refIdxLXB_2, the prediction list utilization flags predFlagLXA_0, predFlagLXA_1, predFlagLXB_0, predFlagLXB_1 and predFlagLXB_2, and the motion vectors mvLXA_0, mvLXA_1, mvLXB_0, mvLXB_1 and mvLXB_2, with X being 0 or 1.
2. The reference indices for the temporal merging candidate, refIdxLXCol, with X being 0 or 1, are set equal to 0.
3. The derivation process for temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked with the luma location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and the variable refIdxL0Col as inputs, and the output being the availability flag availableFlagL0Col and the temporal motion vector mvL0Col.The variables availableFlagCol, predFlagL0Col and predFlagL1Col are derived as follows:

$$\text{availableFlagCol} = \text{availableFlagL0Col} \tag{8-114}$$

$$\text{predFlagL0Col} = \text{availableFlagL0Col} \tag{8-115}$$

$$\text{predFlagL1Col} = 0 \tag{8-116}$$

4. When slice_type is equal to B, the derivation process for temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked with the luma location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and the variable refIdxL1Col as inputs, and the output being the availability flag availableFlagL1Col and the temporal motion vector mvL1Col. The variables availableFlagCol and predFlagL1Col are derived as follows:

$$\text{availableFlagCol} = \text{availableFlagL0Col} \ || \ \text{availableFlagL1Col} \tag{8-117}$$

$$\text{predFlagL1Col} = \text{availableFlagL1Col} \tag{8-118}$$

5. The merging candidate list, mergeCandList, is constructed as follows:

$$\begin{aligned}
&\text{i} = 0\\
&\text{if(availableFlagA}_1\text{)}\\
&\quad \text{mergeCandList[i++]} = \text{A}_1\\
&\text{if(availableFlagB}_1\text{)}\\
&\quad \text{mergeCandList[i++]} = \text{B}_1\\
&\text{if(availableFlagB}_0\text{)}\\
&\quad \text{mergeCandList[i++]} = \text{B}_0\\
&\text{if(availableFlagA}_0\text{)}\\
&\quad \text{mergeCandList[i++]} = \text{A}_0
\end{aligned} \tag{8-119}$$

```
if( availableFlagB2 )
   mergeCandList[ i++ ] = B2
if( availableFlagCol )
   mergeCandList[ i++ ] = Col
```

6. The variable numCurrMergeCand and numOrigMergeCand are set equal to the number of merging candidates in the mergeCandList.

7. When slice_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in clause 8.5.3.2.4 is invoked with mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList, numCurrMergeCand and numOrigMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices $\text{refIdxL0combCand}_k$ and $\text{refIdxL1combCand}_k$, the prediction list utilization flags $\text{predFlagL0combCand}_k$ and $\text{predFlagL1combCand}_k$ and the motion vectors mvL0combCand_k and mvL1combCand_k of every new candidate combCand_k being added into mergeCandList. The number of candidates being added, numCombMergeCand, is set equal to (numCurrMergeCand − numOrigMergeCand). When numCombMergeCand is greater than 0, k ranges from 0 to numCombMergeCand − 1, inclusive.

8. The derivation process for zero motion vector merging candidates specified in clause 8.5.3.2.5 is invoked with the mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList and numCurrMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices $\text{refIdxL0zeroCand}_m$ and $\text{refIdxL1zeroCand}_m$, the prediction list utilization flags $\text{predFlagL0zeroCand}_m$ and $\text{predFlagL1zeroCand}_m$ and the motion vectors mvL0zeroCand_m and mvL1zeroCand_m of every new candidate zeroCand_m being added into mergeCandList. The number of candidates being added, numZeroMergeCand, is set equal to (numCurrMergeCand − numOrigMergeCand − numCombMergeCand). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand − 1, inclusive.

9. The following assignments are made with N being the candidate at position merge_idx[xOrigP][yOrigP] in the merging candidate list mergeCandList (N = mergeCandList[merge_idx[xOrigP][yOrigP]]) and X being replaced by 0 or 1:

$$\text{refIdxLX} = \text{refIdxLXN} \tag{8-120}$$

$$\text{predFlagLX} = \text{predFlagLXN} \tag{8-121}$$

– If use_integer_mv_flag is equal to 0 and the reference picture is not the current picture, the following applies:

$$\text{mvLX}[\ 0\ ] = \text{mvLXN}[\ 0\ ] \tag{8-122}$$

$$\text{mvLX}[\ 1\ ] = \text{mvLXN}[\ 1\ ] \tag{8-123}$$

– Otherwise (use_integer_mv_flag is equal to 1 or the reference picture is the current picture), the following applies:

$$\text{mvLX}[\ 0\ ] = (\ \text{mvLXN}[\ 0\ ]\ \gg\ 2\ )\ \ll\ 2 \tag{8-124}$$

$$\text{mvLX}[\ 1\ ] = (\ \text{mvLXN}[\ 1\ ]\ \gg\ 2\ )\ \ll\ 2 \tag{8-125}$$

10. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1 and (nOrigPbW + nOrigPbH) is equal to 12, the following applies:

$$\text{refIdxL1} = -1 \tag{8-126}$$

$$\text{predFlagL1} = 0 \tag{8-127}$$

8.5.3.2.3 Derivation process for spatial merging candidates

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are as follows, with X being 0 or 1:

– the availability flags availableFlagA$_0$, availableFlagA$_1$, availableFlagB$_0$, availableFlagB$_1$ and availableFlagB$_2$ of the neighbouring prediction units,

– the reference indices refIdxLXA$_0$, refIdxLXA$_1$, refIdxLXB$_0$, refIdxLXB$_1$ and refIdxLXB$_2$ of the neighbouring prediction units,

– the prediction list utilization flags predFlagLXA$_0$, predFlagLXA$_1$, predFlagLXB$_0$, predFlagLXB$_1$ and predFlagLXB$_2$ of the neighbouring prediction units,

– the motion vectors mvLXA$_0$, mvLXA$_1$, mvLXB$_0$, mvLXB$_1$ and mvLXB$_2$ of the neighbouring prediction units.

For the derivation of availableFlagA$_1$, refIdxLXA$_1$, predFlagLXA$_1$ and mvLXA$_1$ the following applies:

– The luma location (xNbA$_1$, yNbA$_1$) inside the neighbouring luma coding block is set equal to (xPb − 1, yPb + nPbH − 1).

– The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbA$_1$, yNbA$_1$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableA$_1$.

– When one or more of the following conditions are true, availableA$_1$ is set equal to FALSE:

  – xPb >> Log2ParMrgLevel is equal to xNbA$_1$ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to yNbA$_1$ >> Log2ParMrgLevel.

  – PartMode of the current prediction unit is equal to PART_Nx2N, PART_nLx2N or PART_nRx2N and partIdx is equal to 1.

– The variables availableFlagA$_1$, refIdxLXA$_1$, predFlagLXA$_1$ and mvLXA$_1$ are derived as follows:

  – If availableA$_1$ is equal to FALSE, availableFlagA$_1$ is set equal to 0, both components of mvLXA$_1$ are set equal to 0, refIdxLXA$_1$ is set equal to −1 and predFlagLXA$_1$ is set equal to 0, with X being 0 or 1.

  – Otherwise, availableFlagA$_1$ is set equal to 1 and the following assignments are made:

$$\text{mvLXA}_1 = \text{MvLX}[\ \text{xNbA}_1\ ][\ \text{yNbA}_1\ ] \quad (8\text{-}128)$$

$$\text{refIdxLXA}_1 = \text{RefIdxLX}[\ \text{xNbA}_1\ ][\ \text{yNbA}_1\ ] \quad (8\text{-}129)$$

$$\text{predFlagLXA}_1 = \text{PredFlagLX}[\ \text{xNbA}_1\ ][\ \text{yNbA}_1\ ] \quad (8\text{-}130)$$

For the derivation of availableFlagB$_1$, refIdxLXB$_1$, predFlagLXB$_1$ and mvLXB$_1$ the following applies:

– The luma location (xNbB$_1$, yNbB$_1$) inside the neighbouring luma coding block is set equal to (xPb + nPbW − 1, yPb − 1).

– The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbB$_1$, yNbB$_1$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableB$_1$.

– When one or more of the following conditions are true, availableB$_1$ is set equal to FALSE:

  – xPb >> Log2ParMrgLevel is equal to xNbB$_1$ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to yNbB$_1$ >> Log2ParMrgLevel.

  – PartMode of the current prediction unit is equal to PART_2NxN, PART_2NxnU or PART_2NxnD and partIdx is equal to 1.

– The variables availableFlagB$_1$, refIdxLXB$_1$, predFlagLXB$_1$ and mvLXB$_1$ are derived as follows:

– If one or more of the following conditions are true, $availableFlagB_1$ is set equal to 0, both components of $mvLXB_1$ are set equal to 0, $refIdxLXB_1$ is set equal to −1 and $predFlagLXB_1$ is set equal to 0, with X being 0 or 1:

  – $availableB_1$ is equal to FALSE.

  – $availableA_1$ is equal to TRUE and the prediction units covering the luma locations ($xNbA_1$, $yNbA_1$) and ($xNbB_1$, $yNbB_1$) have the same motion vectors and the same reference indices.

– Otherwise, $availableFlagB_1$ is set equal to 1 and the following assignments are made:

$$mvLXB_1 = MvLX[\ xNbB_1\ ][\ yNbB_1\ ] \tag{8-131}$$

$$refIdxLXB_1 = RefIdxLX[\ xNbB_1\ ][\ yNbB_1\ ] \tag{8-132}$$

$$predFlagLXB_1 = PredFlagLX[\ xNbB_1\ ][\ yNbB_1\ ] \tag{8-133}$$

For the derivation of $availableFlagB_0$, $refIdxLXB_0$, $predFlagLXB_0$ and $mvLXB_0$ the following applies:

– The luma location ($xNbB_0$, $yNbB_0$) inside the neighbouring luma coding block is set equal to (xPb + nPbW, yPb − 1).

– The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location ($xNbB_0$, $yNbB_0$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag $availableB_0$.

– When xPb >> Log2ParMrgLevel is equal to $xNbB_0$ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to $yNbB_0$ >> Log2ParMrgLevel, $availableB_0$ is set equal to FALSE.

– The variables $availableFlagB_0$, $refIdxLXB_0$, $predFlagLXB_0$ and $mvLXB_0$ are derived as follows:

  – If one or more of the following conditions are true, $availableFlagB_0$ is set equal to 0, both components of $mvLXB_0$ are set equal to 0, $refIdxLXB_0$ is set equal to −1 and $predFlagLXB_0$ is set equal to 0, with X being 0 or 1:

    – $availableB_0$ is equal to FALSE.

    – $availableB_1$ is equal to TRUE and the prediction units covering the luma locations ($xNbB_1$, $yNbB_1$) and ($xNbB_0$, $yNbB_0$) have the same motion vectors and the same reference indices.

  – Otherwise, $availableFlagB_0$ is set equal to 1 and the following assignments are made:

$$mvLXB_0 = MvLX[\ xNbB_0\ ][\ yNbB_0\ ] \tag{8-134}$$

$$refIdxLXB_0 = RefIdxLX[\ xNbB_0\ ][\ yNbB_0\ ] \tag{8-135}$$

$$predFlagLXB_0 = PredFlagLX[\ xNbB_0\ ][\ yNbB_0\ ] \tag{8-136}$$

For the derivation of $availableFlagA_0$, $refIdxLXA_0$, $predFlagLXA_0$ and $mvLXA_0$ the following applies:

– The luma location ($xNbA_0$, $yNbA_0$) inside the neighbouring luma coding block is set equal to (xPb − 1, yPb + nPbH).

– The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location ($xNbA_0$, $yNbA_0$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag $availableA_0$.

– When xPb >> Log2ParMrgLevel is equal to $xNbA_0$ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to yA_0 >> Log2ParMrgLevel, $availableA_0$ is set equal to FALSE.

– The variables $availableFlagA_0$, $refIdxLXA_0$, $predFlagLXA_0$ and $mvLXA_0$ are derived as follows:

  – If one or more of the following conditions are true, $availableFlagA_0$ is set equal to 0, both components of $mvLXA_0$ are set equal to 0, $refIdxLXA_0$ is set equal to −1 and $predFlagLXA_0$ is set equal to 0, with X being 0 or 1:

    – $availableA_0$ is equal to FALSE.

– $availableA_1$ is equal to TRUE and the prediction units covering the luma locations ($xNbA_1$, $yNbA_1$) and ($xNbA_0$, $yNbA_0$) have the same motion vectors and the same reference indices.

– Otherwise, $availableFlagA_0$ is set equal to 1 and the following assignments are made:

$$mvLXA_0 = MvLX[\ xNbA_0\ ][\ yNbA_0\ ] \tag{8-137}$$

$$refIdxLXA_0 = RefIdxLX[\ xNbA_0\ ][\ yNbA_0\ ] \tag{8-138}$$

$$predFlagLXA_0 = PredFlagLX[\ xNbA_0\ ][\ yNbA_0\ ] \tag{8-139}$$

For the derivation of $availableFlagB_2$, $refIdxLXB_2$, $predFlagLXB_2$ and $mvLXB_2$ the following applies:

– The luma location ($xNbB_2$, $yNbB_2$) inside the neighbouring luma coding block is set equal to ($xPb - 1$, $yPb - 1$).

– The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location ($xNbB_2$, $yNbB_2$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag $availableB_2$.

– When xPb >> Log2ParMrgLevel is equal to $xNbB_2$ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to $yNbB_2$ >> Log2ParMrgLevel, $availableB_2$ is set equal to FALSE.

– The variables $availableFlagB_2$, $refIdxLXB_2$, $predFlagLXB_2$ and $mvLXB_2$ are derived as follows:

  – If one or more of the following conditions are true, $availableFlagB_2$ is set equal to 0, both components of $mvLXB_2$ are set equal to 0, $refIdxLXB_2$ is set equal to −1 and $predFlagLXB_2$ is set equal to 0, with X being 0 or 1:

    – $availableB_2$ is equal to FALSE.

    – $availableA_1$ is equal to TRUE and prediction units covering the luma locations ($xNbA_1$, $yNbA_1$) and ($xNbB_2$, $yNbB_2$) have the same motion vectors and the same reference indices.

    – $availableB_1$ is equal to TRUE and the prediction units covering the luma locations ($xNbB_1$, $yNbB_1$) and ($xNbB_2$, $yNbB_2$) have the same motion vectors and the same reference indices.

    – $availableFlagA_0 + availableFlagA_1 + availableFlagB_0 + availableFlagB_1$ is equal to 4.

  – Otherwise, $availableFlagB_2$ is set equal to 1 and the following assignments are made:

$$mvLXB_2 = MvLX[\ xNbB_2\ ][\ yNbB_2\ ] \tag{8-140}$$

$$refIdxLXB_2 = RefIdxLX[\ xNbB_2\ ][\ yNbB_2\ ] \tag{8-141}$$

$$predFlagLXB_2 = PredFlagLX[\ xNbB_2\ ][\ yNbB_2\ ] \tag{8-142}$$

8.5.3.2.4 Derivation process for combined bi-predictive merging candidates

Inputs to this process are:

– a merging candidate list mergeCandList,

– the reference indices refIdxL0N and refIdxL1N of every candidate N in mergeCandList,

– the prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N in mergeCandList,

– the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList,

– the number of elements numCurrMergeCand within mergeCandList,

– the number of elements numOrigMergeCand within the mergeCandList after the spatial and temporal merge candidate derivation process.

Outputs of this process are:

– the merging candidate list mergeCandList,

– the number of elements numCurrMergeCand within mergeCandList,

– the reference indices $refIdxL0combCand_k$ and $refIdxL1combCand_k$ of every new candidate $combCand_k$ added into mergeCandList during the invocation of this process,

– the prediction list utilization flags $predFlagL0combCand_k$ and $predFlagL1combCand_k$ of every new candidate $combCand_k$ added into mergeCandList during the invocation of this process,

– the motion vectors $mvL0combCand_k$ and $mvL1combCand_k$ of every new candidate $combCand_k$ added into mergeCandList during the invocation of this process.

When numOrigMergeCand is greater than 1 and less than MaxNumMergeCand, the variable numInputMergeCand is set equal to numCurrMergeCand, the variable combIdx is set equal to 0, the variable combStop is set equal to FALSE and the following ordered steps are repeated until combStop is equal to TRUE:

1. The variables l0CandIdx and l1CandIdx are derived using combIdx as specified in Table 8-7.
2. The following assignments are made, with l0Cand being the candidate at position l0CandIdx and l1Cand being the candidate at position l1CandIdx in the merging candidate list mergeCandList:
   – l0Cand = mergeCandList[l0CandIdx]
   – l1Cand = mergeCandList[l1CandIdx]
3. When all of the following conditions are true:
   – predFlagL0l0Cand = = 1
   – predFlagL1l1Cand = = 1
   – (DiffPicOrderCnt(RefPicList0[refIdxL0l0Cand], RefPicList1[refIdxL1l1Cand]) != 0) | | (mvL0l0Cand != mvL1l1Cand)

   the candidate $combCand_k$ with k equal to (numCurrMergeCand − numInputMergeCand) is added at the end of mergeCandList, i.e., mergeCandList[numCurrMergeCand] is set equal to $combCand_k$, and the reference indices, the prediction list utilization flags and the motion vectors of $combCand_k$ are derived as follows and numCurrMergeCand is incremented by 1:

$$refIdxL0combCand_k = refIdxL0l0Cand \quad (8\text{-}143)$$

$$refIdxL1combCand_k = refIdxL1l1Cand \quad (8\text{-}144)$$

$$predFlagL0combCand_k = 1 \quad (8\text{-}145)$$

$$predFlagL1combCand_k = 1 \quad (8\text{-}146)$$

$$mvL0combCand_k[\ 0\ ] = mvL0l0Cand[\ 0\ ] \quad (8\text{-}147)$$

$$mvL0combCand_k[\ 1\ ] = mvL0l0Cand[\ 1\ ] \quad (8\text{-}148)$$

$$mvL1combCand_k[\ 0\ ] = mvL1l1Cand[\ 0\ ] \quad (8\text{-}149)$$

$$mvL1combCand_k[\ 1\ ] = mvL1l1Cand[\ 1\ ] \quad (8\text{-}150)$$

$$numCurrMergeCand = numCurrMergeCand + 1 \quad (8\text{-}151)$$

4. The variable combIdx is incremented by 1.
5. When combIdx is equal to (numOrigMergeCand * (numOrigMergeCand − 1)) or numCurrMergeCand is equal to MaxNumMergeCand, combStop is set equal to TRUE.

Table 8-7 – Specification of l0CandIdx and l1CandIdx

combIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
l0CandIdx	0	1	0	2	1	2	0	3	1	3	2	3
l1CandIdx	1	0	2	0	2	1	3	0	3	1	3	2

8.5.3.2.5 Derivation process for zero motion vector merging candidates

Inputs to this process are:

– a merging candidate list mergeCandList,

– the reference indices refIdxL0N and refIdxL1N of every candidate N in mergeCandList,

– the prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N in mergeCandList,

– the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList,

– the number of elements numCurrMergeCand within mergeCandList.

Outputs of this process are:

– the merging candidate list mergeCandList,

– the number of elements numCurrMergeCand within mergeCandList,

– the reference indices $\text{refIdxL0zeroCand}_m$ and $\text{refIdxL1zeroCand}_m$ of every new candidate zeroCand_m added into mergeCandList during the invocation of this process,

– the prediction list utilization flags $\text{predFlagL0zeroCand}_m$ and $\text{predFlagL1zeroCand}_m$ of every new candidate zeroCand_m added into mergeCandList during the invocation of this process,

– the motion vectors mvL0zeroCand_m and mvL1zeroCand_m of every new candidate zeroCand_m added into mergeCandList during the invocation of this process.

The variable numRefIdx is derived as follows:

– If slice_type is equal to P, numRefIdx is set equal to num_ref_idx_l0_active_minus1 + 1.

– Otherwise (slice_type is equal to B), numRefIdx is set equal to Min(num_ref_idx_l0_active_minus1 + 1, num_ref_idx_l1_active_minus1 + 1).

When numCurrMergeCand is less than MaxNumMergeCand, the variable numInputMergeCand is set equal to numCurrMergeCand, the variable zeroIdx is set equal to 0 and the following ordered steps are repeated until numCurrMergeCand is equal to MaxNumMergeCand:

1. For the derivation of the reference indices, the prediction list utilization flags and the motion vectors of the zero motion vector merging candidate, the following applies:

   – If slice_type is equal to P, the candidate zeroCand_m with m equal to (numCurrMergeCand − numInputMergeCand) is added at the end of mergeCandList, i.e., mergeCandList[numCurrMergeCand] is set equal to zeroCand_m, and the reference indices, the prediction list utilization flags and the motion vectors of zeroCand_m are derived as follows and numCurrMergeCand is incremented by 1:

$$\text{refIdxL0zeroCand}_m = (\ \text{zeroIdx} < \text{numRefIdx}\ )\ ?\ \text{zeroIdx} : 0 \quad (8\text{-}152)$$

$$\text{refIdxL1zeroCand}_m = -1 \quad (8\text{-}153)$$

$$\text{predFlagL0zeroCand}_m = 1 \quad (8\text{-}154)$$

$$\text{predFlagL1zeroCand}_m = 0 \quad (8\text{-}155)$$

$$\text{mvL0zeroCand}_m[\ 0\ ] = 0 \quad (8\text{-}156)$$

$$\text{mvL0zeroCand}_m[\ 1\ ] = 0 \quad (8\text{-}157)$$

$$\text{mvL1zeroCand}_m[\ 0\ ] = 0 \quad (8\text{-}158)$$

$$\text{mvL1zeroCand}_m[\ 1\ ] = 0 \quad (8\text{-}159)$$

$$\text{numCurrMergeCand} = \text{numCurrMergeCand} + 1 \quad (8\text{-}160)$$

   – Otherwise (slice_type is equal to B), the candidate zeroCand_m with m equal to (numCurrMergeCand − numInputMergeCand) is added at the end of mergeCandList, i.e.,

mergeCandList[numCurrMergeCand] is set equal to $zeroCand_m$, and the reference indices, the prediction list utilization flags and the motion vectors of $zeroCand_m$ are derived as follows and numCurrMergeCand is incremented by 1:

$$refIdxL0zeroCand_m = (zeroIdx < numRefIdx) ? zeroIdx : 0 \quad (8\text{-}161)$$

$$refIdxL1zeroCand_m = (zeroIdx < numRefIdx) ? zeroIdx : 0 \quad (8\text{-}162)$$

$$predFlagL0zeroCand_m = 1 \quad (8\text{-}163)$$

$$predFlagL1zeroCand_m = 1 \quad (8\text{-}164)$$

$$mvL0zeroCand_m[0] = 0 \quad (8\text{-}165)$$

$$mvL0zeroCand_m[1] = 0 \quad (8\text{-}166)$$

$$mvL1zeroCand_m[0] = 0 \quad (8\text{-}167)$$

$$mvL1zeroCand_m[1] = 0 \quad (8\text{-}168)$$

$$numCurrMergeCand = numCurrMergeCand + 1 \quad (8\text{-}169)$$

2. The variable zeroIdx is incremented by 1.

8.5.3.2.6 Derivation process for luma motion vector prediction

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– the reference index of the current prediction unit partition refIdxLX, with X being 0 or 1,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Output of this process is the prediction mvpLX of the motion vector mvLX, with X being 0 or 1.

The motion vector predictor mvpLX is derived in the following ordered steps:

1. The derivation process for motion vector predictor candidates from neighbouring prediction unit partitions in clause 8.5.3.2.7 is invoked with the luma coding block location (xCb, yCb), the coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, refIdxLX, with X being 0 or 1 and the partition index partIdx as inputs, and the availability flags availableFlagLXN and the motion vectors mvLXN, with N being replaced by A or B, as output.

2. If both availableFlagLXA and availableFlagLXB are equal to 1 and mvLXA is not equal to mvLXB, availableFlagLXCol is set equal to 0. Otherwise, the derivation process for temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked with luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and refIdxLX, with X being 0 or 1 as inputs, and with the output being the availability flag availableFlagLXCol and the temporal motion vector predictor mvLXCol.

3. The motion vector predictor candidate list, mvpListLX, is constructed as follows:

```
i = 0
if( availableFlagLXA ) {
   mvpListLX[ i++ ] = mvLXA
   if( availableFlagLXB && ( mvLXA != mvLXB ) )
      mvpListLX[ i++ ] = mvLXB
} else if( availableFlagLXB )
   mvpListLX[ i++ ] = mvLXB                                        (8-170)
```

```
if( i < 2  &&  availableFlagLXCol )
    mvpListLX[ i++ ] = mvLXCol
while( i < 2 ) {
    mvpListLX[ i ][ 0 ] = 0
    mvpListLX[ i ][ 1 ] = 0
    i++
}
```

4. The motion vector of mvpListLX[mvp_lX_flag[xPb][yPb]] is assigned to mvpLX.

8.5.3.2.7 Derivation process for motion vector predictor candidates

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– the reference index of the current prediction unit partition refIdxLX, with X being 0 or 1,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are (with N being replaced by A or B):

– the motion vectors mvLXN of the neighbouring prediction units,

– the availability flags availableFlagLXN of the neighbouring prediction units.

Figure 8-3 provides an overview of spatial motion vector neighbours.

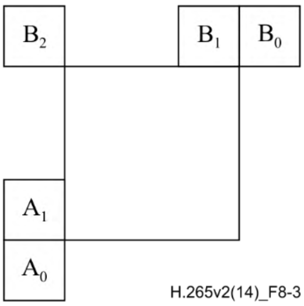


Figure 8-3 – Spatial motion vector neighbours (informative)

The variable currPb specifies the current luma prediction block at luma location (xPb, yPb) and the variable currPic specifies the current picture.

The variable isScaledFlagLX, with X being 0 or 1, is set equal to 0.

The motion vector mvLXA and the availability flag availableFlagLXA are derived in the following ordered steps:

1. The sample location ($xNbA_0$, $yNbA_0$) is set equal to (xPb − 1, yPb + nPbH) and the sample location ($xNbA_1$, $yNbA_1$) is set equal to ($xNbA_0$, $yNbA_0$ − 1).

2. The availability flag availableFlagLXA is set equal to 0 and both components of mvLXA are set equal to 0.

3. The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbY, yNbY) set equal to ($xNbA_0$, $yNbA_0$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag $availableA_0$.

4. The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb),

the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbY, yNbY) set equal to ($xNbA_1$, $yNbA_1$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag $availableA_1$.

5. When $availableA_0$ or $availableA_1$ is equal to TRUE, the variable isScaledFlagLX is set equal to 1.

6. The following applies for ($xNbA_k$, $yNbA_k$) from ($xNbA_0$, $yNbA_0$) to ($xNbA_1$, $yNbA_1$):

   – When $availableA_k$ is equal to TRUE and availableFlagLXA is equal to 0, the following applies:

     – If PredFlagLX[$xNbA_k$][$yNbA_k$] is equal to 1 and DiffPicOrderCnt(RefPicListX[RefIdxLX[$xNbA_k$][$yNbA_k$]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXA is set equal to 1 and the following applies:

$$mvLXA = MvLX[\ xNbA_k\ ][\ yNbA_k\ ] \tag{8-171}$$

     – Otherwise, when PredFlagLY[$xNbA_k$][$yNbA_k$] (with Y = !X) is equal to 1 and DiffPicOrderCnt(RefPicListY[RefIdxLY[$xNbA_k$][$yNbA_k$]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXA is set equal to 1 and the following applies:

$$mvLXA = MvLY[\ xNbA_k\ ][\ yNbA_k\ ] \tag{8-172}$$

7. When availableFlagLXA is equal to 0, the following applies for ($xNbA_k$, $yNbA_k$) from ($xNbA_0$, $yNbA_0$) to ($xNbA_1$, $yNbA_1$) or until availableFlagLXA is equal to 1:

   – When $availableA_k$ is equal to TRUE and availableFlagLXA is equal to 0, the following applies:

     – If PredFlagLX[$xNbA_k$][$yNbA_k$] is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLX[$xNbA_k$][$yNbA_k$], RefPicListX), availableFlagLXA is set equal to 1 and the following assignments are made:

$$mvLXA = MvLX[\ xNbA_k\ ][\ yNbA_k\ ] \tag{8-173}$$

$$refIdxA = RefIdxLX[\ xNbA_k\ ][\ yNbA_k\ ] \tag{8-174}$$

$$refPicListA = RefPicListX \tag{8-175}$$

     – Otherwise, when PredFlagLY[$xNbA_k$][$yNbA_k$] (with Y = !X) is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLY[$xNbA_k$][$yNbA_k$], RefPicListY), availableFlagLXA is set equal to 1 and the following assignments are made:

$$mvLXA = MvLY[\ xNbA_k\ ][\ yNbA_k\ ] \tag{8-176}$$

$$refIdxA = RefIdxLY[\ xNbA_k\ ][\ yNbA_k\ ] \tag{8-177}$$

$$refPicListA = RefPicListY \tag{8-178}$$

   – When availableFlagLXA is equal to 1, DiffPicOrderCnt(refPicListA[refIdxA], RefPicListX[refIdxLX]) is not equal to 0, and both refPicListA[refIdxA] and RefPicListX[refIdxLX] are short-term reference pictures, mvLXA is derived as follows:

$$tx = (\ 16384 + (\ Abs(\ td\ )\ >>\ 1\ )\ )\ /\ td \tag{8-179}$$

$$distScaleFactor = Clip3(\ -4096,\ 4095,\ (\ tb * tx + 32\ )\ >>\ 6\ ) \tag{8-180}$$

$$\begin{aligned} mvLXA = Clip3(\ &-32768,\ 32767,\ Sign(\ distScaleFactor * mvLXA\ ) * \\ &(\ (\ Abs(\ distScaleFactor * mvLXA\ ) + 127\ )\ >>\ 8\ )\ ) \end{aligned} \tag{8-181}$$

   where td and tb are derived as follows:

$$td = Clip3(\ -128,\ 127,\ DiffPicOrderCnt(\ currPic,\ refPicListA[\ refIdxA\ ]\ )\ ) \tag{8-182}$$

$$tb = Clip3(\ -128,\ 127,\ DiffPicOrderCnt(\ currPic,\ RefPicListX[\ refIdxLX\ ]\ )\ ) \tag{8-183}$$

The motion vector mvLXB and the availability flag availableFlagLXB are derived in the following ordered steps:

1. The sample locations ($xNbB_0$, $yNbB_0$), ($xNbB_1$, $yNbB_1$) and ($xNbB_2$, $yNbB_2$) are set equal to (xPb + nPbW, yPb − 1), (xPb + nPbW − 1, yPb − 1) and (xPb − 1, yPb − 1), respectively.

2. The availability flag availableFlagLXB is set equal to 0 and the both components of mvLXB are set equal to 0.

3. The following applies for ($xNbB_k$, $yNbB_k$) from ($xNbB_0$, $yNbB_0$) to ($xNbB_2$, $yNbB_2$):

   – The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbY, yNbY) set equal to ($xNbB_k$, $yNbB_k$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag $availableB_k$.

   – When $availableB_k$ is equal to TRUE and availableFlagLXB is equal to 0, the following applies:

     – If PredFlagLX[$xNbB_k$][$yNbB_k$] is equal to 1, and DiffPicOrderCnt(RefPicListX[RefIdxLX[$xNbB_k$][$yNbB_k$]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXB is set equal to 1 and the following assignment are made:

$$mvLXB = MvLX[\ xNbB_k\ ][\ yNbB_k\ ] \qquad (8\text{-}184)$$

     – Otherwise, when PredFlagLY[$xNbB_k$][$yNbB_k$] (with Y = !X) is equal to 1 and DiffPicOrderCnt(RefPicListY[RefIdxLY[$xNbB_k$][$yNbB_k$]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXB is set equal to 1 and the following assignment is made:

$$mvLXB = MvLY[\ xNbB_k\ ][\ yNbB_k\ ] \qquad (8\text{-}185)$$

4. When isScaledFlagLX is equal to 0 and availableFlagLXB is equal to 1, availableFlagLXA is set equal to 1 and the following applies:

$$mvLXA = mvLXB \qquad (8\text{-}186)$$

5. When isScaledFlagLX is equal to 0, availableFlagLXB is set equal to 0 and the following applies for ($xNbB_k$, $yNbB_k$) from ($xNbB_0$, $yNbB_0$) to ($xNbB_2$, $yNbB_2$) or until availableFlagLXB is equal to 1:

   – The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbY, yNbY) set equal to ($xNbB_k$, $yNbB_k$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag $availableB_k$.

   – When $availableB_k$ is equal to TRUE and availableFlagLXB is equal to 0, the following applies:

     – If PredFlagLX[$xNbB_k$][$yNbB_k$] is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLX[$xNbB_k$][$yNbB_k$], RefPicListX), availableFlagLXB is set equal to 1 and the following assignments are made:

$$mvLXB = MvLX[\ xNbB_k\ ][\ yNbB_k\ ] \qquad (8\text{-}187)$$

$$refIdxB = RefIdxLX[\ xNbB_k\ ][\ yNbB_k\ ] \qquad (8\text{-}188)$$

$$refPicListB = RefPicListX \qquad (8\text{-}189)$$

     – Otherwise, when PredFlagLY[$xNbB_k$][$yNbB_k$] (with Y = !X) is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLY[$xNbB_k$][$yNbB_k$], RefPicListY), availableFlagLXB is set equal to 1 and the following assignments are made:

$$mvLXB = MvLY[\ xNbB_k\ ][\ yNbB_k\ ] \qquad (8\text{-}190)$$

$$refIdxB = RefIdxLY[\ xNbB_k\ ][\ yNbB_k\ ] \qquad (8\text{-}191)$$

$$refPicListB = RefPicListY \qquad (8\text{-}192)$$

– When availableFlagLXB is equal to 1, DiffPicOrderCnt(refPicListB[refIdxB], RefPicListX[refIdxLX]) is not equal to 0 and both refPicListB[refIdxB] and RefPicListX[refIdxLX] are short-term reference pictures, mvLXB is derived as follows:

$$tx = (16384 + (Abs(td) >> 1)) / td \qquad (8\text{-}193)$$

$$distScaleFactor = Clip3(-4096, 4095, (tb * tx + 32) >> 6) \qquad (8\text{-}194)$$

$$mvLXB = Clip3(-32768, 32767, Sign(distScaleFactor * mvLXB) * ((Abs(distScaleFactor * mvLXB) + 127) >> 8)) \qquad (8\text{-}195)$$

where td and tb are derived as follows:

$$td = Clip3(-128, 127, DiffPicOrderCnt(currPic, refPicListB[refIdxB])) \qquad (8\text{-}196)$$

$$tb = Clip3(-128, 127, DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])) \qquad (8\text{-}197)$$

8.5.3.2.8 Derivation process for temporal luma motion vector prediction

Inputs to this process are:

– a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

– the motion vector prediction mvLXCol,

– the availability flag availableFlagLXCol.

The variable currPb specifies the current luma prediction block at luma location (xPb, yPb).

The variables mvLXCol and availableFlagLXCol are derived as follows:

– If slice_temporal_mvp_enabled_flag is equal to 0, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

– Otherwise (slice_temporal_mvp_enabled_flag is equal to 1), the following ordered steps apply:

1. The bottom right collocated motion vector is derived as follows:

$$xColBr = xPb + nPbW \qquad (8\text{-}198)$$

$$yColBr = yPb + nPbH \qquad (8\text{-}199)$$

   – If yPb >> CtbLog2SizeY is equal to yColBr >> CtbLog2SizeY, yColBr is less than pic_height_in_luma_samples and xColBr is less than pic_width_in_luma_samples, the following applies:

     – The variable colPb specifies the luma prediction block covering the modified location given by ((xColBr >> 4) << 4, (yColBr >> 4) << 4) inside the collocated picture specified by ColPic.

     – The luma location (xColPb, yColPb) is set equal to the top-left sample of the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic.

     – The derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

   – Otherwise, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

2. When availableFlagLXCol is equal to 0, the central collocated motion vector is derived as follows:

$$xColCtr = xPb + (nPbW >> 1) \qquad (8\text{-}200)$$

$$\text{yColCtr} = \text{yPb} + (\text{ nPbH} >> 1) \tag{8-201}$$

– The variable colPb specifies the luma prediction block covering the modified location given by ((xColCtr >> 4) << 4, (yColCtr >> 4) << 4) inside the collocated picture specified by ColPic.

– The luma location (xColPb, yColPb) is set equal to the top-left sample of the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic.

– The derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

8.5.3.2.9 Derivation process for collocated motion vectors

Inputs to this process are:

– a variable currPb specifying the current prediction block,

– a variable colPb specifying the collocated prediction block inside the collocated picture specified by ColPic,

– a luma location (xColPb, yColPb) specifying the top-left sample of the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic,

– a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

– the motion vector prediction mvLXCol,

– the availability flag availableFlagLXCol.

The variable currPic specifies the current picture.

The arrays predFlagL0Col[x][y], mvL0Col[x][y] and refIdxL0Col[x][y] are set equal to PredFlagL0[x][y], MvL0[x][y] and RefIdxL0[x][y], respectively, of the collocated picture specified by ColPic, and the arrays predFlagL1Col[x][y], mvL1Col[x][y] and refIdxL1Col[x][y] are set equal to PredFlagL1[x][y], MvL1[x][y] and RefIdxL1[x][y], respectively, of the collocated picture specified by ColPic.

The variables mvLXCol and availableFlagLXCol are derived as follows:

– If colPb is coded in an intra prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

– Otherwise, the motion vector mvCol, the reference index refIdxCol and the reference list identifier listCol are derived as follows:

  – If predFlagL0Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol and listCol are set equal to mvL1Col[xColPb][yColPb], refIdxL1Col[xColPb][yColPb] and L1, respectively.

  – Otherwise, if predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol and listCol are set equal to mvL0Col[xColPb][yColPb], refIdxL0Col[xColPb][yColPb] and L0, respectively.

  – Otherwise (predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 1), the following assignments are made:

    – If NoBackwardPredFlag is equal to 1, mvCol, refIdxCol and listCol are set equal to mvLXCol[xColPb][yColPb], refIdxLXCol[xColPb][yColPb] and LX, respectively.

    – Otherwise, mvCol, refIdxCol and listCol are set equal to mvLNCol[xColPb][yColPb], refIdxLNCol[xColPb][yColPb] and LN, respectively, with N being the value of collocated_from_l0_flag.

  And mvLXCol and availableFlagLXCol are derived as follows:

  – If LongTermRefPic(currPic, currPb, refIdxLX, LX) is not equal to LongTermRefPic(ColPic, colPb, refIdxCol, listCol), both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

  – Otherwise, the variable availableFlagLXCol is set equal to 1, refPicListCol[refIdxCol] is set to be the picture with reference index refIdxCol in the reference picture list listCol of the slice containing prediction block colPb in the collocated picture specified by ColPic, and the following applies:

$$\text{colPocDiff} = \text{DiffPicOrderCnt(ColPic, refPicListCol[refIdxCol])} \tag{8-202}$$

$$\text{currPocDiff} = \text{DiffPicOrderCnt}(\ \text{currPic},\ \text{RefPicListX}[\ \text{refIdxLX}\ ]\ ) \tag{8-203}$$

– If RefPicListX[refIdxLX] is a long-term reference picture, or colPocDiff is equal to currPocDiff, mvLXCol is derived as follows:

$$\text{mvLXCol} = \text{mvCol} \tag{8-204}$$

– Otherwise, mvLXCol is derived as a scaled version of the motion vector mvCol as follows:

$$\text{tx} = (\ 16384 + (\ \text{Abs}(\ \text{td}\ ) >> 1\ )\ )\ /\ \text{td} \tag{8-205}$$

$$\text{distScaleFactor} = \text{Clip3}(\ -4096,\ 4095,\ (\ \text{tb} * \text{tx} + 32\ ) >> 6\ ) \tag{8-206}$$

$$\begin{aligned}\text{mvLXCol} = {} & \text{Clip3}(\ -32768,\ 32767,\ \text{Sign}(\ \text{distScaleFactor} * \text{mvCol}\ ) * \\ & (\ (\ \text{Abs}(\ \text{distScaleFactor} * \text{mvCol}\ ) + 127\ ) >> 8\ )\ )\end{aligned} \tag{8-207}$$

where td and tb are derived as follows:

$$\text{td} = \text{Clip3}(\ -128,\ 127,\ \text{colPocDiff}\ ) \tag{8-208}$$

$$\text{tb} = \text{Clip3}(\ -128,\ 127,\ \text{currPocDiff}\ ) \tag{8-209}$$

8.5.3.2.10 Derivation process for chroma motion vectors

This process is invoked when ChromaArrayType is not equal to 0.

Input to this process is a luma motion vector mvLX.

Output of this process is a chroma motion vector mvCLX.

A chroma motion vector is derived from the corresponding luma motion vector.

For the derivation of the chroma motion vector mvCLX, the following applies:

$$\text{mvCLX}[\ 0\ ] = \text{mvLX}[\ 0\ ] * 2\ /\ \text{SubWidthC} \tag{8-210}$$

$$\text{mvCLX}[\ 1\ ] = \text{mvLX}[\ 1\ ] * 2\ /\ \text{SubHeightC} \tag{8-211}$$

8.5.3.3 Decoding process for inter prediction samples

8.5.3.3.1 General

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– the luma motion vectors mvL0 and mvL1,

– when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags, predFlagL0, and predFlagL1.

Outputs of this process are:

– an $(\text{nCbS}_L)\text{x}(\text{nCbS}_L)$ array predSamples_L of luma prediction samples, where nCbS_L is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(\text{nCbSw}_C)\text{x}(\text{nCbSh}_C)$ array predSamples_{Cb} of chroma prediction samples for the component Cb, where nCbSw_C and nCbSh_C are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to nCbS. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to nCbS / SubWidthC and the variable $nCbSh_C$ is set equal to nCbS / SubHeightC.

Let $predSamplesL0_L$ and $predSamplesL1_L$ be (nPbW)x(nPbH) arrays of predicted luma sample values and, when ChromaArrayType is not equal to 0, $predSamplesL0_{Cb}$, $predSamplesL1_{Cb}$, $predSamplesL0_{Cr}$ and $predSamplesL1_{Cr}$ be (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays of predicted chroma sample values.

For X being each of 0 and 1, when predFlagLX is equal to 1, the following applies:

– The reference picture consisting of an ordered two-dimensional array $refPicLX_L$ of luma samples and, when ChromaArrayType is not equal to 0, two ordered two-dimensional arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.3.3.2 with refIdxLX as input.

– The array $predSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$ are derived by invoking the fractional sample interpolation process specified in clause 8.5.3.3.3 with the luma locations (xCb, yCb) and (xBl, yBl), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors mvLX and, when ChromaArrayType is not equal to 0, mvCLX, and the reference arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cr}$ as inputs.

The prediction samples inside the current luma prediction block, $predSamples_L[x_L + xBl][y_L + yBl]$ with $x_L = 0..nPbW - 1$ and $y_L = 0..nPbH - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width nPbW, the prediction block height nPbH and the sample arrays $predSamplesL0_L$ and $predSamplesL1_L$, and the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1 and cIdx equal to 0 as inputs.

When ChromaArrayType is not equal to 0, the prediction samples inside the current chroma component Cb prediction block, $predSamples_{Cb}[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC]$ with $x_C = 0..nPbW / SubWidthC - 1$ and $y_C = 0..nPbH / SubHeightC - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width nPbW set equal to nPbW / SubWidthC, the prediction block height nPbH set equal to nPbH / SubHeightC, the sample arrays $predSamplesL0_{Cb}$ and $predSamplesL1_{Cb}$, and the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1 and cIdx equal to 1 as inputs.

When ChromaArrayType is not equal to 0, the prediction samples inside the current chroma component Cr prediction block, $predSamples_{Cr}[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC]$ with $x_C = 0..nPbW / SubWidthC - 1$ and $y_C = 0..nPbH / SubHeightC - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width nPbW set equal to nPbW / SubWidthC, the prediction block height nPbH set equal to nPbH / SubHeightC, the sample arrays $predSamplesL0_{Cr}$ and $predSamplesL1_{Cr}$, and the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1 and cIdx equal to 2 as inputs.

8.5.3.3.2 Reference picture selection process

Input to this process is a reference index refIdxLX.

Output of this process is a reference picture consisting of a two-dimensional array of luma samples $refPicLX_L$ and, when ChromaArrayType is not equal to 0, two two-dimensional arrays of chroma samples $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

The output reference picture RefPicListX[refIdxLX] consists of a pic_width_in_luma_samples by pic_height_in_luma_samples array of luma samples $refPicLX_L$ and, when ChromaArrayType is not equal to 0, two PicWidthInSamplesC by PicHeightInSamplesC arrays of chroma samples $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

The reference picture sample arrays $refPicLX_L$, $refPicLX_{Cb}$ and $refPicLX_{Cr}$ correspond to decoded sample arrays S_L, S_{Cb} and S_{Cr} derived in clause 8.7 for a previously-decoded picture.

8.5.3.3.3 Fractional sample interpolation process

8.5.3.3.3.1 General

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture

– a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block

– a luma motion vector mvLX given in quarter-luma-sample units

– when ChromaArrayType is not equal to 0, a chroma motion vector mvCLX given in eighth-chroma-sample units

– the selected reference picture sample array $refPicLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

Outputs of this process are:

– an (nPbW)x(nPbH) array $predSamplesLX_L$ of prediction luma sample values

– when ChromaArrayType is not equal to 0, two (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$ of prediction chroma sample values.

The location (xPb, yPb) given in full-sample units of the upper-left luma samples of the current prediction block relative to the upper-left luma sample location of the given reference sample arrays is derived as follows:

$$xPb = xCb + xBl \quad (8\text{-}212)$$

$$yPb = yCb + yBl \quad (8\text{-}213)$$

Let ($xInt_L$, $yInt_L$) be a luma location given in full-sample units and ($xFrac_L$, $yFrac_L$) be an offset given in quarter-sample units. These variables are used only in this clause for specifying fractional-sample locations inside the reference sample arrays $refPicLX_L$, $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

For each luma sample location (x_L = 0..nPbW − 1, y_L = 0..nPbH − 1) inside the prediction luma sample array $predSamplesLX_L$, the corresponding prediction luma sample value $predSamplesLX_L[x_L][y_L]$ is derived as follows:

– The variables $xInt_L$, $yInt_L$, $xFrac_L$ and $yFrac_L$ are derived as follows:

$$xInt_L = xPb + (mvLX[0] >> 2) + x_L \quad (8\text{-}214)$$

$$yInt_L = yPb + (mvLX[1] >> 2) + y_L \quad (8\text{-}215)$$

$$xFrac_L = mvLX[0] \& 3 \quad (8\text{-}216)$$

$$yFrac_L = mvLX[1] \& 3 \quad (8\text{-}217)$$

– The prediction luma sample value $predSamplesLX_L[x_L][y_L]$ is derived by invoking the process specified in clause 8.5.3.3.3.2 with ($xInt_L$, $yInt_L$), ($xFrac_L$, $yFrac_L$) and $refPicLX_L$ as inputs.

When ChromaArrayType is not equal to 0, the following applies.

Let ($xInt_C$, $yInt_C$) be a chroma location given in full-sample units and ($xFrac_C$, $yFrac_C$) be an offset given in one-eighth sample units. These variables are used only in this clause for specifying general fractional-sample locations inside the reference sample arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

For each chroma sample location (x_C = 0..nPbW / SubWidthC − 1, y_C = 0..nPbH / SubHeightC − 1) inside the prediction chroma sample arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$, the corresponding prediction chroma sample values $predSamplesLX_{Cb}[x_C][y_C]$ and $predSamplesLX_{Cr}[x_C][y_C]$ are derived as follows:

– The variables $xInt_C$, $yInt_C$, $xFrac_C$ and $yFrac_C$ are derived as follows:

$$xInt_C = (xPb / SubWidthC) + (mvCLX[0] >> 3) + x_C \quad (8\text{-}218)$$

$$yInt_C = (yPb / SubHeightC) + (mvCLX[1] >> 3) + y_C \quad (8\text{-}219)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8\text{-}220)$$

$$yFrac_C = mvCLX[1] \& 7 \quad (8\text{-}221)$$

– The prediction sample value $predSamplesLX_{Cb}[x_C][y_C]$ is derived by invoking the process specified in clause 8.5.3.3.3.3 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and $refPicLX_{Cb}$ as inputs.

– The prediction sample value $predSamplesLX_{Cr}[x_C][y_C]$ is derived by invoking the process specified in clause 8.5.3.3.3.3 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and $refPicLX_{Cr}$ as inputs.

8.5.3.3.3.2 Luma sample interpolation process

Inputs to this process are:

– a luma location in full-sample units ($xInt_L$, $yInt_L$),

– a luma location in fractional-sample units ($xFrac_L$, $yFrac_L$),

– the luma reference sample array $refPicLX_L$.

Output of this process is a predicted luma sample value $predSampleLX_L$

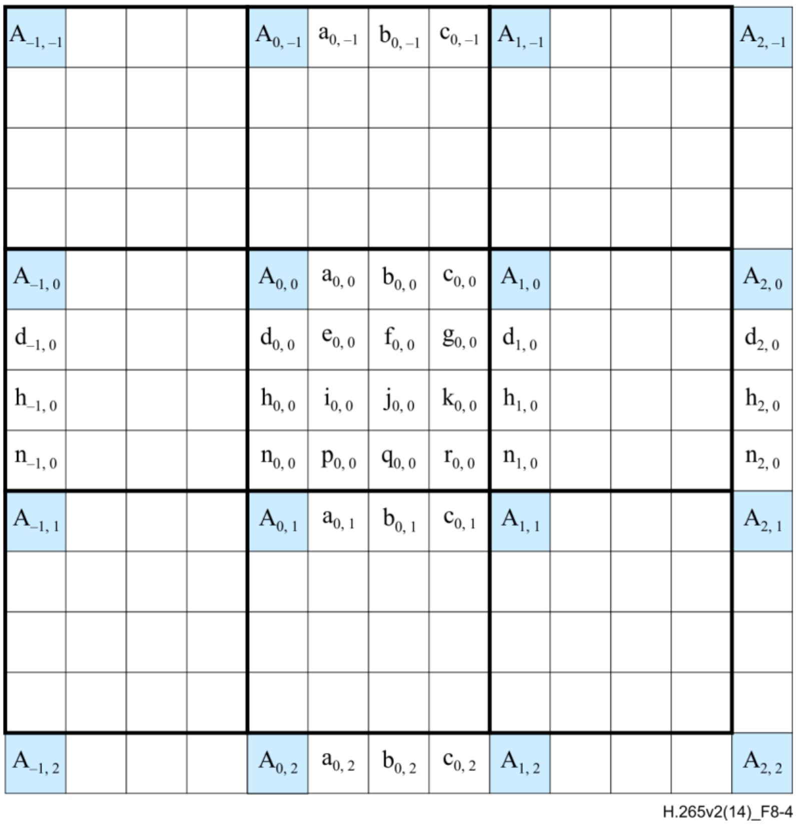


Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation

In Figure 8-4, the positions labelled with upper-case letters $A_{i,j}$ within shaded blocks represent luma samples at full-sample locations inside the given two-dimensional array $refPicLX_L$ of luma samples. These samples may be used for generating the predicted luma sample value $predSampleLX_L$. The locations ($xA_{i,j}$, $yA_{i,j}$) for each of the corresponding luma samples $A_{i,j}$ inside the given array $refPicLX_L$ of luma samples are derived as follows:

$$xA_{i,j} = \text{Clip3}(0, \text{pic\_width\_in\_luma\_samples} - 1, xInt_L + i) \quad (8\text{-}222)$$

$$yA_{i,j} = \text{Clip3}(0, \text{pic\_height\_in\_luma\_samples} - 1, yInt_L + j) \quad (8\text{-}223)$$

The positions labelled with lower-case letters within un-shaded blocks represent luma samples at quarter-luma-sample fractional locations. The luma location offset in fractional-sample units ($xFrac_L$, $yFrac_L$) specifies which of the generated luma samples at full-sample and fractional-sample locations is assigned to the predicted luma sample value $predSampleLX_L$. This assignment is as specified in Table 8-8. The value of $predSampleLX_L$ is the output.

The variables shift1, shift2 and shift3 are derived as follows:

– The variable shift1 is set equal to Min(4, $BitDepth_Y - 8$), the variable shift2 is set equal to 6 and the variable shift3 is set equal to Max(2, $14 - BitDepth_Y$).

Given the luma samples $A_{i,j}$ at full-sample locations ($xA_{i,j}$, $yA_{i,j}$), the luma samples $a_{0,0}$ to $r_{0,0}$ at fractional sample positions are derived as follows:

– The samples labelled $a_{0,0}$, $b_{0,0}$, $c_{0,0}$, $d_{0,0}$, $h_{0,0}$ and $n_{0,0}$ are derived by applying an 8-tap filter to the nearest integer position samples as follows:

$$a_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 10 * A_{-1,0} + 58 * A_{0,0} + 17 * A_{1,0} - 5 * A_{2,0} + A_{3,0}) >> \text{shift1} \quad (8\text{-}224)$$

$$b_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 11 * A_{-1,0} + 40 * A_{0,0} + 40 * A_{1,0} - 11 * A_{2,0} + 4 * A_{3,0} - A_{4,0}) >> \text{shift1} \quad (8\text{-}225)$$

$$c_{0,0} = (A_{-2,0} - 5 * A_{-1,0} + 17 * A_{0,0} + 58 * A_{1,0} - 10 * A_{2,0} + 4 * A_{3,0} - A_{4,0}) >> \text{shift1} \quad (8\text{-}226)$$

$$d_{0,0} = (-A_{0,-3} + 4 * A_{0,-2} - 10 * A_{0,-1} + 58 * A_{0,0} + 17 * A_{0,1} - 5 * A_{0,2} + A_{0,3}) >> \text{shift1} \quad (8\text{-}227)$$

$$h_{0,0} = (-A_{0,-3} + 4 * A_{0,-2} - 11 * A_{0,-1} + 40 * A_{0,0} + 40 * A_{0,1} - 11 * A_{0,2} + 4 * A_{0,3} - A_{0,4}) >> \text{shift1} \quad (8\text{-}228)$$

$$n_{0,0} = (A_{0,-2} - 5 * A_{0,-1} + 17 * A_{0,0} + 58 * A_{0,1} - 10 * A_{0,2} + 4 * A_{0,3} - A_{0,4}) >> \text{shift1} \quad (8\text{-}229)$$

– The samples labelled $e_{0,0}$, $i_{0,0}$, $p_{0,0}$, $f_{0,0}$, $j_{0,0}$, $q_{0,0}$, $g_{0,0}$, $k_{0,0}$ and $r_{0,0}$ are derived by applying an 8-tap filter to the samples $a_{0,i}$, $b_{0,i}$ and $c_{0,i}$ with $i = -3..4$ in the vertical direction as follows:

$$e_{0,0} = (-a_{0,-3} + 4 * a_{0,-2} - 10 * a_{0,-1} + 58 * a_{0,0} + 17 * a_{0,1} - 5 * a_{0,2} + a_{0,3}) >> \text{shift2} \quad (8\text{-}230)$$

$$i_{0,0} = (-a_{0,-3} + 4 * a_{0,-2} - 11 * a_{0,-1} + 40 * a_{0,0} + 40 * a_{0,1} - 11 * a_{0,2} + 4 * a_{0,3} - a_{0,4}) >> \text{shift2} \quad (8\text{-}231)$$

$$p_{0,0} = (a_{0,-2} - 5 * a_{0,-1} + 17 * a_{0,0} + 58 * a_{0,1} - 10 * a_{0,2} + 4 * a_{0,3} - a_{0,4}) >> \text{shift2} \quad (8\text{-}232)$$

$$f_{0,0} = (-b_{0,-3} + 4 * b_{0,-2} - 10 * b_{0,-1} + 58 * b_{0,0} + 17 * b_{0,1} - 5 * b_{0,2} + b_{0,3}) >> \text{shift2} \quad (8\text{-}233)$$

$$j_{0,0} = (-b_{0,-3} + 4 * b_{0,-2} - 11 * b_{0,-1} + 40 * b_{0,0} + 40 * b_{0,1} - 11 * b_{0,2} + 4 * b_{0,3} - b_{0,4}) >> \text{shift2} \quad (8\text{-}234)$$

$$q_{0,0} = (b_{0,-2} - 5 * b_{0,-1} + 17 * b_{0,0} + 58 * b_{0,1} - 10 * b_{0,2} + 4 * b_{0,3} - b_{0,4}) >> \text{shift2} \quad (8\text{-}235)$$

$$g_{0,0} = (-c_{0,-3} + 4 * c_{0,-2} - 10 * c_{0,-1} + 58 * c_{0,0} + 17 * c_{0,1} - 5 * c_{0,2} + c_{0,3}) >> \text{shift2} \quad (8\text{-}236)$$

$$k_{0,0} = (-c_{0,-3} + 4 * c_{0,-2} - 11 * c_{0,-1} + 40 * c_{0,0} + 40 * c_{0,1} - 11 * c_{0,2} + 4 * c_{0,3} - c_{0,4}) >> \text{shift2} \quad (8\text{-}237)$$

$$r_{0,0} = (c_{0,-2} - 5 * c_{0,-1} + 17 * c_{0,0} + 58 * c_{0,1} - 10 * c_{0,2} + 4 * c_{0,3} - c_{0,4}) >> \text{shift2} \quad (8\text{-}238)$$

Table 8-8 – Assignment of the luma prediction sample predSampleLX$_L$

xFracL	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
yFracL	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
predSampleLX$_L$	A << shift3	d	h	n	a	e	i	p	b	f	j	q	c	g	k	r

8.5.3.3.3.3 Chroma sample interpolation process

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

– a chroma location in full-sample units ($xInt_C$, $yInt_C$),

– a chroma location in eighth fractional-sample units ($xFrac_C$, $yFrac_C$),

– the chroma reference sample array $refPicLX_C$.

Output of this process is a predicted chroma sample value $predSampleLX_C$

	$ha_{0,-1}$	$hb_{0,-1}$	$hc_{0,-1}$	$hd_{0,-1}$	$he_{0,-1}$	$hf_{0,-1}$	$hg_{0,-1}$	$hh_{0,-1}$	
$ah_{-1,0}$	$B_{0,0}$	$ab_{0,0}$	$ac_{0,0}$	$ad_{0,0}$	$ae_{0,0}$	$af_{0,0}$	$ag_{0,0}$	$ah_{0,0}$	$B_{1,0}$
$bh_{-1,0}$	$ba_{0,0}$	$bb_{0,0}$	$bc_{0,0}$	$bd_{0,0}$	$be_{0,0}$	$bf_{0,0}$	$bg_{0,0}$	$bh_{0,0}$	$ba_{1,0}$
$ch_{-1,0}$	$ca_{0,0}$	$cb_{0,0}$	$cc_{0,0}$	$cd_{0,0}$	$ce_{0,0}$	$cf_{0,0}$	$cg_{0,0}$	$ch_{0,0}$	$ca_{1,0}$
$dh_{-1,0}$	$da_{0,0}$	$db_{0,0}$	$dc_{0,0}$	$dd_{0,0}$	$de_{0,0}$	$df_{0,0}$	$dg_{0,0}$	$dh_{0,0}$	$da_{1,0}$
$eh_{-1,0}$	$ea_{0,0}$	$eb_{0,0}$	$ec_{0,0}$	$ed_{0,0}$	$ee_{0,0}$	$ef_{0,0}$	$eg_{0,0}$	$eh_{0,0}$	$ea_{1,0}$
$fh_{-1,0}$	$fa_{0,0}$	$fb_{0,0}$	$fc_{0,0}$	$fd_{0,0}$	$fe_{0,0}$	$ff_{0,0}$	$fg_{0,0}$	$fh_{0,0}$	$fa_{1,0}$
$gh_{-1,0}$	$ga_{0,0}$	$gb_{0,0}$	$gc_{0,0}$	$gd_{0,0}$	$ge_{0,0}$	$gf_{0,0}$	$gg_{0,0}$	$gh_{0,0}$	$ga_{1,0}$
$hh_{-1,0}$	$ha_{0,0}$	$hb_{0,0}$	$hc_{0,0}$	$hd_{0,0}$	$he_{0,0}$	$hf_{0,0}$	$hg_{0,0}$	$hh_{0,0}$	$ha_{1,0}$
	$B_{0,1}$	$ab_{0,1}$	$ac_{0,1}$	$ad_{0,1}$	$ae_{0,1}$	$af_{0,1}$	$ag_{0,1}$	$ah_{0,1}$	$B_{1,1}$

H.265v2(14)_F8-5

Figure 8-5 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for eighth sample chroma interpolation

In Figure 8-5, the positions labelled with upper-case letters $B_{i,j}$ within shaded blocks represent chroma samples at full-sample locations inside the given two-dimensional array $refPicLX_C$ of chroma samples. These samples may be used for generating the predicted chroma sample value $predSampleLX_C$. The locations ($xB_{i,j}$, $yB_{i,j}$) for each of the corresponding chroma samples $B_{i,j}$ inside the given array $refPicLX_C$ of chroma samples are derived as follows:

$$xB_{i,j} = \text{Clip3}(0, (pic\_width\_in\_luma\_samples / SubWidthC) - 1, xInt_C + i) \quad (8\text{-}239)$$

$$yB_{i,j} = \text{Clip3}(0, (pic\_height\_in\_luma\_samples / SubHeightC) - 1, yInt_C + j) \quad (8\text{-}240)$$

The positions labelled with lower-case letters within un-shaded blocks represent chroma samples at eighth-pel sample fractional locations. The chroma location offset in fractional-sample units ($xFrac_C$, $yFrac_C$) specifies which of the generated chroma samples at full-sample and fractional-sample locations is assigned to the predicted chroma sample value $predSampleLX_C$. This assignment is as specified in Table 8-9. The output is the value of $predSampleLX_C$.

The variables shift1, shift2 and shift3 are derived as follows:

– The variable shift1 is set equal to Min(4, $BitDepth_C - 8$), the variable shift2 is set equal to 6 and the variable shift3 is set equal to Max(2, $14 - BitDepth_C$).

Given the chroma samples $B_{i,j}$ at full-sample locations ($xB_{i,j}$, $yB_{i,j}$), the chroma samples $ab_{0,0}$ to $hh_{0,0}$ at fractional sample positions are derived as follows:

– The samples labelled $ab_{0,0}$, $ac_{0,0}$, $ad_{0,0}$, $ae_{0,0}$, $af_{0,0}$, $ag_{0,0}$ and $ah_{0,0}$ are derived by applying a 4-tap filter to the nearest integer position samples as follows:

$$ab_{0,0} = (-2 * B_{-1,0} + 58 * B_{0,0} + 10 * B_{1,0} - 2 * B_{2,0}) >> shift1 \quad (8\text{-}241)$$

$$ac_{0,0} = (-4 * B_{-1,0} + 54 * B_{0,0} + 16 * B_{1,0} - 2 * B_{2,0}) >> shift1 \quad (8\text{-}242)$$

$$ad_{0,0} = (-6 * B_{-1,0} + 46 * B_{0,0} + 28 * B_{1,0} - 4 * B_{2,0}) >> shift1 \quad (8\text{-}243)$$

$$ae_{0,0} = (-4 * B_{-1,0} + 36 * B_{0,0} + 36 * B_{1,0} - 4 * B_{2,0}) >> shift1 \quad (8\text{-}244)$$

$$af_{0,0} = (-4 * B_{-1,0} + 28 * B_{0,0} + 46 * B_{1,0} - 6 * B_{2,0}) >> shift1 \quad (8\text{-}245)$$

$$ag_{0,0} = (-2 * B_{-1,0} + 16 * B_{0,0} + 54 * B_{1,0} - 4 * B_{2,0}) >> shift1 \quad (8\text{-}246)$$

$$ah_{0,0} = (-2 * B_{-1,0} + 10 * B_{0,0} + 58 * B_{1,0} - 2 * B_{2,0}) >> shift1 \quad (8\text{-}247)$$

– The samples labelled $ba_{0,0}$, $ca_{0,0}$, $da_{0,0}$, $ea_{0,0}$, $fa_{0,0}$, $ga_{0,0}$ and $ha_{0,0}$ are derived by applying a 4-tap filter to the nearest integer position samples as follows:

$$ba_{0,0} = (-2 * B_{0,-1} + 58 * B_{0,0} + 10 * B_{0,1} - 2 * B_{0,2}) >> shift1 \quad (8\text{-}248)$$

$$ca_{0,0} = (-4 * B_{0,-1} + 54 * B_{0,0} + 16 * B_{0,1} - 2 * B_{0,2}) >> shift1 \quad (8\text{-}249)$$

$$da_{0,0} = (-6 * B_{0,-1} + 46 * B_{0,0} + 28 * B_{0,1} - 4 * B_{0,2}) >> shift1 \quad (8\text{-}250)$$

$$ea_{0,0} = (-4 * B_{0,-1} + 36 * B_{0,0} + 36 * B_{0,1} - 4 * B_{0,2}) >> shift1 \quad (8\text{-}251)$$

$$fa_{0,0} = (-4 * B_{0,-1} + 28 * B_{0,0} + 46 * B_{0,1} - 6 * B_{0,2}) >> shift1 \quad (8\text{-}252)$$

$$ga_{0,0} = (-2 * B_{0,-1} + 16 * B_{0,0} + 54 * B_{0,1} - 4 * B_{0,2}) >> shift1 \quad (8\text{-}253)$$

$$ha_{0,0} = (-2 * B_{0,-1} + 10 * B_{0,0} + 58 * B_{0,1} - 2 * B_{0,2}) >> shift1 \quad (8\text{-}254)$$

– The samples labelled $bX_{0,0}$, $cX_{0,0}$, $dX_{0,0}$, $eX_{0,0}$, $fX_{0,0}$, $gX_{0,0}$ and $hX_{0,0}$ for X being replaced by b, c, d, e, f, g and h, respectively, are derived by applying a 4-tap filter to the intermediate values $aX_{0,i}$ with i = −1..2 in the vertical direction as follows:

$$bX_{0,0} = (-2 * aX_{0,-1} + 58 * aX_{0,0} + 10 * aX_{0,1} - 2 * aX_{0,2}) >> shift2 \quad (8\text{-}255)$$

$$cX_{0,0} = (-4 * aX_{0,-1} + 54 * aX_{0,0} + 16 * aX_{0,1} - 2 * aX_{0,2}) >> shift2 \quad (8\text{-}256)$$

$$dX_{0,0} = (-6 * aX_{0,-1} + 46 * aX_{0,0} + 28 * aX_{0,1} - 4 * aX_{0,2}) >> shift2 \quad (8\text{-}257)$$

$$eX_{0,0} = (-4 * aX_{0,-1} + 36 * aX_{0,0} + 36 * aX_{0,1} - 4 * aX_{0,2}) >> shift2 \quad (8\text{-}258)$$

$$fX_{0,0} = (-4 * aX_{0,-1} + 28 * aX_{0,0} + 46 * aX_{0,1} - 6 * aX_{0,2}) >> shift2 \quad (8\text{-}259)$$

$$gX_{0,0} = (-2 * aX_{0,-1} + 16 * aX_{0,0} + 54 * aX_{0,1} - 4 * aX_{0,2}) >> shift2 \quad (8\text{-}260)$$

$$hX_{0,0} = (-2 * aX_{0,-1} + 10 * aX_{0,0} + 58 * aX_{0,1} - 2 * aX_{0,2}) >> shift2 \quad (8\text{-}261)$$

Table 8-9 – Assignment of the chroma prediction sample predSampleLX$_C$ for (X, Y) being replaced by (1, b), (2, c), (3, d), (4, e), (5, f), (6, g) and (7, h), respectively

xFracC	0	0	0	0	0	0	0	0
yFracC	0	1	2	3	4	5	6	7
predSampleLX$_C$	B << shift3	ba	ca	da	ea	fa	ga	ha
xFracC	X	X	X	X	X	X	X	X
yFracC	0	1	2	3	4	5	6	7
predSampleLX$_C$	aY	bY	cY	dY	eY	fY	gY	hY

8.5.3.3.4 Weighted sample prediction process

8.5.3.3.4.1 General

Inputs to this process are:

– two variables nPbW and nPbH specifying the width and the height of the current prediction block,

– two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1,

– the prediction list utilization flags, predFlagL0 and predFlagL1,

– the reference indices refIdxL0 and refIdxL1,

– a variable cIdx specifying colour component index.

Output of this process is the (nPbW)x(nPbH) array pbSamples of prediction sample values.

The variable bitDepth is derived as follows:

– If cIdx is equal to 0, bitDepth is set equal to BitDepth_Y.

– Otherwise, bitDepth is set equal to BitDepth_C.

The variable weightedPredFlag is derived as follows:

– If slice_type is equal to P, weightedPredFlag is set equal to weighted_pred_flag.

– Otherwise (slice_type is equal to B), weightedPredFlag is set equal to weighted_bipred_flag.

The following applies:

– If weightedPredFlag is equal to 0, the array pbSamples of the prediction samples is derived by invoking the default weighted sample prediction process as specified in clause 8.5.3.3.4.2 with the prediction block width nPbW, the prediction block height nPbH, two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1, the prediction list utilization flags predFlagL0 and predFlagL1 and the bit depth bitDepth as inputs.

– Otherwise (weightedPredFlag is equal to 1), the array pbSamples of the prediction samples is derived by invoking the weighted sample prediction process as specified in clause 8.5.3.3.4.3 with the prediction block width nPbW, the prediction block height nPbH, two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1, the prediction list utilization flags predFlagL0 and predFlagL1, the reference indices refIdxL0 and refIdxL1, the colour component index cIdx and the bit depth bitDepth as inputs.

8.5.3.3.4.2 Default weighted sample prediction process

Inputs to this process are:

– two variables nPbW and nPbH specifying the width and the height of the current prediction block,

– two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1,

– the prediction list utilization flags, predFlagL0, and predFlagL1,

– a bit depth of samples, bitDepth.

Output of this process is the (nPbW)x(nPbH) array pbSamples of prediction sample values.

Variables shift1, shift2, offset1 and offset2 are derived as follows:

– The variable shift1 is set equal to Max(2, 14 − bitDepth) and the variable shift2 is set equal to Max(3, 15 − bitDepth).

– The variable offset1 is set equal to 1 << (shift1 − 1).

– The variable offset2 is set equal to 1 << (shift2 − 1).

Depending on the values of predFlagL0 and predFlagL1, the prediction samples pbSamples[x][y] with x = 0..nPbW − 1 and y = 0..nPbH − 1 are derived as follows:

– If predFlagL0 is equal to 1 and predFlagL1 is equal to 0, the prediction sample values are derived as follows:

$$\text{pbSamples}[\,x\,][\,y\,] = \text{Clip3}(\,0,\,(\,1 \ll \text{bitDepth}\,) - 1,\,(\,\text{predSamplesL0}[\,x\,][\,y\,] + \text{offset1}\,) \gg \text{shift1}\,) \quad (8\text{-}262)$$

– Otherwise, if predFlagL0 is equal to 0 and predFlagL1 is equal to 1, the prediction sample values are derived as follows:

$$\text{pbSamples}[\,x\,][\,y\,] = \text{Clip3}(\,0,\,(\,1 \ll \text{bitDepth}\,) - 1,\,(\,\text{predSamplesL1}[\,x\,][\,y\,] + \text{offset1}\,) \gg \text{shift1}\,) \quad (8\text{-}263)$$

– Otherwise (predFlagL0 is equal to 1 and predFlagL1 is equal to 1), the prediction sample values are derived as follows:

$$\begin{aligned}\text{pbSamples}[\,x\,][\,y\,] = \text{Clip3}(\,&0,\,(\,1 \ll \text{bitDepth}\,) - 1,\\ &(\,\text{predSamplesL0}[\,x\,][\,y\,] + \text{predSamplesL1}[\,x\,][\,y\,] + \text{offset2}\,) \gg \text{shift2}\,)\end{aligned} \quad (8\text{-}264)$$

8.5.3.3.4.3 Explicit weighted sample prediction process

Inputs to this process are:

– two variables nPbW and nPbH specifying the width and the height of the current prediction block,
– two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1,
– the prediction list utilization flags, predFlagL0 and predFlagL1,
– the reference indices, refIdxL0 and refIdxL1,
– a variable cIdx specifying colour component index,
– a bit depth of samples, bitDepth.

Output of this process is the (nPbW)x(nPbH) array pbSamples of prediction sample values.

The variable shift1 is set equal to Max(2, 14 − bitDepth).

The variables log2Wd, o0, o1, w0 and w1 are derived as follows:

– If cIdx is equal to 0 for luma samples, the following applies:

$$\text{log2Wd} = \text{luma\_log2\_weight\_denom} + \text{shift1} \tag{8-265}$$

$$\text{w0} = \text{LumaWeightL0[refIdxL0]} \tag{8-266}$$

$$\text{w1} = \text{LumaWeightL1[refIdxL1]} \tag{8-267}$$

$$\text{o0} = \text{luma\_offset\_l0[refIdxL0]} \ll \text{WpOffsetBdShift}_Y \tag{8-268}$$

$$\text{o1} = \text{luma\_offset\_l1[refIdxL1]} \ll \text{WpOffsetBdShift}_Y \tag{8-269}$$

– Otherwise (cIdx is not equal to 0 for chroma samples), the following applies:

$$\text{log2Wd} = \text{ChromaLog2WeightDenom} + \text{shift1} \tag{8-270}$$

$$\text{w0} = \text{ChromaWeightL0[refIdxL0][cIdx} - 1\text{]} \tag{8-271}$$

$$\text{w1} = \text{ChromaWeightL1[refIdxL1][cIdx} - 1\text{]} \tag{8-272}$$

$$\text{o0} = \text{ChromaOffsetL0[refIdxL0][cIdx} - 1\text{]} \ll \text{WpOffsetBdShift}_C \tag{8-273}$$

$$\text{o1} = \text{ChromaOffsetL1[refIdxL1][cIdx} - 1\text{]} \ll \text{WpOffsetBdShift}_C \tag{8-274}$$

The prediction sample pbSamples[x][y] with x = 0..nPbW − 1 and y = 0..nPbH − 1 are derived as follows:

– If the predFlagL0 is equal to 1 and predFlagL1 is equal to 0, the prediction sample values are derived as follows:

$$\begin{aligned}&\text{pbSamples[x][y]} = \text{Clip3(0, (1} \ll \text{bitDepth)} - 1,\\&\quad\text{((predSamplesL0[x][y] * w0} + 2^{\text{log2Wd} - 1}\text{)} \gg \text{log2Wd)} + \text{o0)}\end{aligned} \tag{8-275}$$

– Otherwise, if the predFlagL0 is equal to 0 and predFlagL1 is equal to 1, the prediction sample values are derived as follows:

$$\begin{aligned}&\text{pbSamples[x][y]} = \text{Clip3(0, (1} \ll \text{bitDepth)} - 1,\\&\quad\text{((predSamplesL1[x][y] * w1} + 2^{\text{log2Wd} - 1}\text{)} \gg \text{log2Wd)} + \text{o1)}\end{aligned} \tag{8-276}$$

– Otherwise (predFlagL0 is equal to 1 and predFlagL1 is equal to 1), the prediction sample values are derived as follows:

$$\begin{aligned}&\text{pbSamples[x][y]} = \text{Clip3(0, (1} \ll \text{bitDepth)} - 1,\\&\quad\text{(predSamplesL0[x][y] * w0} + \text{predSamplesL1[x][y] * w1} +\\&\quad\quad\text{((o0} + \text{o1} + 1\text{)} \ll \text{log2Wd))} \gg \text{(log2Wd} + 1\text{))}\end{aligned} \tag{8-277}$$

8.5.4 Decoding process for the residual signal of coding units coded in inter prediction mode

8.5.4.1 General

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block.

Outputs of this process are:

– an $(nCbS_L)x(nCbS_L)$ array $resSamples_L$ of luma residual samples, where $nCbS_L$ is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $resSamples_{Cb}$ of chroma residual samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $resSamples_{Cr}$ of chroma residual samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to 1 << log2CbSize. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS_L$ / SubWidthC and the variable $nCbSh_C$ is set equal to $nCbS_L$ / SubHeightC.

Let $resSamples_L$ be an $(nCbS_L)x(nCbS_L)$ array of luma residual samples and, when ChromaArrayType is not equal to 0, let $resSamples_{Cb}$ and $resSamples_{Cr}$ be two $(nCbSw_C)x(nCbSh_C)$ arrays of chroma residual samples.

Depending on the value of rqt_root_cbf, the following applies:

– If rqt_root_cbf is equal to 0 or cu_skip_flag[xCb][yCb] is equal to 1, all samples of the $(nCbS_L)x(nCbS_L)$ array $resSamples_L$ and, when ChromaArrayType is not equal to 0, all samples of the two $(nCbSw_C)x(nCbSh_C)$ arrays $resSamples_{Cb}$ and $resSamples_{Cr}$ are set equal to 0.

– Otherwise (rqt_root_cbf is equal to 1), the following ordered steps apply:

1. The decoding process for luma residual blocks as specified in clause 8.5.4.2 below is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable nCbS set equal to $nCbS_L$ and the $(nCbS_L)x(nCbS_L)$ array $resSamples_L$ as inputs, and the output is a modified version of the $(nCbS_L)x(nCbS_L)$ array $resSamples_L$.

2. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 1, the variable nCbSw set equal to $nCbSw_C$, the variable nCbSh set equal to $nCbSh_C$ and the $(nCbSw_C)x(nCbSh_C)$ array $resSamples_{Cb}$ as inputs, and the output is a modified version of the $(nCbSw_C)x(nCbSh_C)$ array $resSamples_{Cb}$.

3. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 2, the variable nCbSw set equal to $nCbSw_C$, the variable nCbSh set equal to $nCbSh_C$ and the $(nCbSw_C)x(nCbSh_C)$ array $resSamples_{Cr}$ as inputs, and the output is a modified version of the $(nCbSw_C)x(nCbSh_C)$ array $resSamples_{Cr}$.

4. When residual_adaptive_colour_transform_enabled_flag is equal to 1, the residual modification process for blocks using adaptive colour transform as specified in clause 8.6.8 is invoked with location (xCb, yCb), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable resSampleArrayL set equal to $resSamples_L$, the variable resSampleArrayCb set equal to $resSamples_{Cb}$ and the variable resSampleArrayCr set equal to $resSamples_{Cr}$ as inputs, and the outputs are modified versions of $resSample_L$, $resSample_{Cb}$ and $resSample_{Cr}$.

8.5.4.2 Decoding process for luma residual blocks

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xB0, yB0) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,

– a variable log2TrafoSize specifying the size of the current luma block,

– a variable trafoDepth specifying the hierarchy depth of the current luma block relative to the luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– an (nCbS)x(nCbS) array resSamples of luma residual samples.

Output of this process is a modified version of the (nCbS)x(nCbS) array of luma residual samples.

Depending on the value of split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth], the following applies:

– If split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth] is equal to 1, the following ordered steps apply:

1. The variables xB1 and yB1 are derived as follows:
   – The variable xB1 is set equal to xB0 + (1 << (log2TrafoSize − 1)).
   – The variable yB1 is set equal to yB0 + (1 << (log2TrafoSize − 1)).
2. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCbS and the (nCbS)x(nCbS) array resSamples as inputs, and the output is a modified version of the (nCbS)x(nCbS) array resSamples.
3. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB1, yB0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCbS and the (nCbS)x(nCbS) array resSamples as inputs, and the output is a modified version of the (nCbS)x(nCbS) array resSamples.
4. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB0, yB1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCbS and the (nCbS)x(nCbS) array resSamples as inputs, and the output is a modified version of the (nCbS)x(nCbS) array resSamples.
5. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB1, yB1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCbS and the (nCbS)x(nCbS) array resSamples as inputs, and the output is a modified version of the (nCbS)x(nCbS) array resSamples.

– Otherwise (split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth] is equal to 0), the following ordered steps apply:

1. The variable nTbS is set equal to 1 << log2TrafoSize.
2. The scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location (xCb + xB0, yCb + yB0), the variable trafoDepth, the variable cIdx set equal to 0 and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array transformBlock.
3. When explicit_rdpcm_flag[xCb + xB0][yCb + yB0][0] is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable mDir set equal to explicit_rdpcm_dir_flag[xCb + xB0][yCb + yB0][0], the variable nTbS and the (nTbS)x(nTbS) array r set equal to the array transformBlock as inputs, and the output is a modified (nTbS)x(nTbS) array transformBlock.
4. The (nCbS)x(nCbS) residual sample array of the current coding block resSamples is modified as follows:

$$\text{resSamples}[\ xB0 + i,\ yB0 + j\ ] = \text{transformBlock}[\ i,\ j\ ],\ \text{with } i = 0..\text{nTbS} - 1,\ j = 0..\text{nTbS} - 1 \quad (8\text{-}278)$$

8.5.4.3 Decoding process for chroma residual blocks

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
– a luma location (xB0, yB0) specifying the top-left luma sample of the current chroma block relative to the top-left sample of the current luma coding block,
– a variable log2TrafoSize specifying the size of the current chroma block in luma samples,
– a variable trafoDepth specifying the hierarchy depth of the current chroma block relative to the chroma coding block,
– a variable cIdx specifying the chroma component of the current block,
– the variables nCbSw and nCbSh specifying the width and height, respectively, of the current chroma coding block,
– an (nCbSw)x(nCbSh) array resSamples of chroma residual samples.

Output of this process is a modified version of the (nCbSw)x(nCbSh) array of chroma residual samples.

The variable splitChromaFlag is derived as follows:

– If split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth] is equal to 1 and one or more of the following conditions are met, splitChromaFlag is set equal to 1:

  – log2TrafoSize is greater than 3.

  – ChromaArrayType is equal to 3.

– Otherwise (split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth] is equal to 0 or both log2TrafoSize is equal to 3 and ChromaArrayType is not equal to 3), splitChromaFlag is set equal to 0.

Depending on the value of splitChromaFlag, the following applies:

– If splitChromaFlag is equal to 1, the following ordered steps apply:

  1. The variables xB1 and yB1 are derived as follows:

     – The variable xB1 is set equal to xB0 + (1 << (log2TrafoSize − 1)).

     – The variable yB1 is set equal to yB0 + (1 << (log2TrafoSize − 1)).

  2. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.

  3. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB1, yB0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.

  4. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB0, yB1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.

  5. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB1, yB1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.

– Otherwise (splitChromaFlag is equal to 0), for the variable blkIdx proceeding over the values 0..(ChromaArrayType = = 2 ? 1 : 0), the following ordered steps apply:

  1. The variable nTbS is set equal to (1 << log2TrafoSize) / SubWidthC.

  2. The variable yBN is set equal to yB0 + blkIdx * nTbS * SubHeightC.

  3. The scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location (xCb + xB0, yCb + yBN), the variable trafoDepth, the variable cIdx and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array transformBlock.

  4. When explicit_rdpcm_flag[xCb + xB0][yCb + yBN][cIdx] is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable mDir set equal to explicit_rdpcm_dir_flag[xCb + xB0][yCb + yBN][cIdx], the variable nTbS and the (nTbS)x(nTbS) array r set equal to the array transformBlock as inputs, and the output is a modified (nTbS)x(nTbS) array transformBlock.

  5. When cross_component_prediction_enabled_flag is equal to 1 and ChromaArrayType is equal to 3, the residual modification process for transform blocks using cross-component prediction as specified in clause 8.6.6 is invoked with the transform block location (xCb + xB0, yCb + yB0), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array r_Y set equal to the corresponding luma residual sample array transformBlock of the current transform block and the (nTbS)x(nTbS) array r set equal to the array transformBlock as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.

6. The (nCbS)x(nCbS) residual sample array of the current coding block resSamples is modified as follows, for i = 0..nTbS − 1, j = 0..nTbS − 1:

   resSamples[(xCb + xB0) / SubWidthC + i, (yCb + yBN) / SubHeightC + j] =
   transformBlock[i, j] (8-279)

8.6 Scaling, transformation and array construction process prior to deblocking filter process

8.6.1 Derivation process for quantization parameters

Input to this process is a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture.

In this process, the variable Qp_Y, the luma quantization parameter Qp'_Y and the chroma quantization parameters Qp'_{Cb} and Qp'_{Cr} are derived.

The luma location (xQg, yQg), specifies the top-left luma sample of the current quantization group relative to the top-left luma sample of the current picture. The horizontal and vertical positions xQg and yQg are set equal to xCb − (xCb & ((1 << Log2MinCuQpDeltaSize) − 1)) and yCb − (yCb & ((1 << Log2MinCuQpDeltaSize) − 1)), respectively. The luma size of a quantization group, Log2MinCuQpDeltaSize, determines the luma size of the smallest area inside a CTB that shares the same qP_{Y_PRED}.

The predicted luma quantization parameter qP_{Y_PRED} is derived by the following ordered steps:

1. The variable qP_{Y_PREV} is derived as follows:
   – If one or more of the following conditions are true, qP_{Y_PREV} is set equal to $SliceQp_Y$:
     – The current quantization group is the first quantization group in a slice.
     – The current quantization group is the first quantization group in a tile.
     – The current quantization group is the first quantization group in a CTB row of a tile and entropy_coding_sync_enabled_flag is equal to 1.
   – Otherwise, qP_{Y_PREV} is set equal to the luma quantization parameter Qp_Y of the last coding unit in the previous quantization group in decoding order.
2. The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring location (xNbY, yNbY) set equal to (xQg − 1, yQg) as inputs, and the output is assigned to availableA. The variable qP_{Y_A} is derived as follows:
   – If one or more of the following conditions are true, qP_{Y_A} is set equal to qP_{Y_PREV}:
     – availableA is equal to FALSE.
     – the CTB address ctbAddrA of the CTB containing the luma coding block covering the luma location (xQg − 1, yQg) is not equal to CtbAddrInTs, where ctbAddrA is derived as follows:

       xTmp = (xQg − 1) >> MinTbLog2SizeY
       yTmp = yQg >> MinTbLog2SizeY
       minTbAddrA = MinTbAddrZs[xTmp][yTmp]
       ctbAddrA = minTbAddrA >> (2 * (CtbLog2SizeY − MinTbLog2SizeY)) (8-280)

   – Otherwise, qP_{Y_A} is set equal to the luma quantization parameter Qp_Y of the coding unit containing the luma coding block covering (xQg − 1, yQg).
3. The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring location (xNbY, yNbY) set equal to (xQg, yQg − 1) as inputs, and the output is assigned to availableB. The variable qP_{Y_B} is derived as follows:
   – If one or more of the following conditions are true, qP_{Y_B} is set equal to qP_{Y_PREV}:
     – availableB is equal to FALSE.
     – the CTB address ctbAddrB of the CTB containing the luma coding block covering the luma location (xQg, yQg − 1) is not equal to CtbAddrInTs, where ctbAddrB is derived as follows:

       xTmp = xQg >> MinTbLog2SizeY
       yTmp = (yQg − 1) >> MinTbLog2SizeY

$$\text{minTbAddrB} = \text{MinTbAddrZs}[\ \text{xTmp}\ ][\ \text{yTmp}\ ]$$
$$\text{ctbAddrB} = \text{minTbAddrB}\ \gg\ (\ 2 * (\ \text{CtbLog2SizeY} - \text{MinTbLog2SizeY}\ )\ ) \qquad (8\text{-}281)$$

– Otherwise, qP_{Y_B} is set equal to the luma quantization parameter Qp_Y of the coding unit containing the luma coding block covering (xQg, yQg − 1).

4. The predicted luma quantization parameter qP_{Y_PRED} is derived as follows:

$$qP_{Y\_PRED} = (\ qP_{Y\_A} + qP_{Y\_B} + 1\ )\ \gg\ 1 \qquad (8\text{-}282)$$

The variable Qp_Y is derived as follows:

$$Qp_Y = (\ (\ qP_{Y\_PRED} + \text{CuQpDeltaVal} + 52 + 2 * \text{QpBdOffset}_Y\ )\%(\ 52 + \text{QpBdOffset}_Y\ )\ ) - \text{QpBdOffset}_Y \qquad (8\text{-}283)$$

The luma quantization parameter Qp'_Y is derived as follows:

$$Qp'_Y = Qp_Y + \text{QpBdOffset}_Y \qquad (8\text{-}284)$$

When ChromaArrayType is not equal to 0, the following applies:

– The variables qP_{Cb} and qP_{Cr} are derived as follows:

– If tu_residual_act_flag[xTbY][yTbY] is equal to 0, the following applies:

$$qPi_{Cb} = \text{Clip3}(\ -\text{QpBdOffset}_C,\ 57,\ Qp_Y + \text{pps\_cb\_qp\_offset} + \text{slice\_cb\_qp\_offset} + \text{CuQpOffset}_{Cb}\ ) \qquad (8\text{-}285)$$

$$qPi_{Cr} = \text{Clip3}(\ -\text{QpBdOffset}_C,\ 57,\ Qp_Y + \text{pps\_cr\_qp\_offset} + \text{slice\_cr\_qp\_offset} + \text{CuQpOffset}_{Cr}\ ) \qquad (8\text{-}286)$$

– Otherwise (tu_residual_act_flag[xTbY][yTbY] is equal to 1), the following applies:

$$qPi_{Cb} = \text{Clip3}(\ -\text{QpBdOffsetC},\ 57,\ \text{QpY} + \text{PpsActQpOffsetCb} + \text{slice\_act\_cb\_qp\_offset} + \text{CuQpOffsetCb}\ ) \qquad (8\text{-}287)$$

$$qPi_{Cr} = \text{Clip3}(\ -\text{QpBdOffsetC},\ 57,\ \text{QpY} + \text{PpsActQpOffsetCr} + \text{slice\_act\_cr\_qp\_offset} + \text{CuQpOffsetCr}\ ) \qquad (8\text{-}288)$$

– If ChromaArrayType is equal to 1, the variables qP_{Cb} and qP_{Cr} are set equal to the value of Qp_C as specified in Table 8-10 based on the index qPi equal to qPi_{Cb} and qPi_{Cr}, respectively.

– Otherwise, the variables qP_{Cb} and qP_{Cr} are set equal to Min(qPi, 51), based on the index qPi equal to qPi_{Cb} and qPi_{Cr}, respectively.

– The chroma quantization parameters for the Cb and Cr components, Qp'_{Cb} and Qp'_{Cr}, are derived as follows:

$$Qp'_{Cb} = qP_{Cb} + \text{QpBdOffset}_C \qquad (8\text{-}289)$$

$$Qp'_{Cr} = qP_{Cr} + \text{QpBdOffset}_C \qquad (8\text{-}290)$$

Table 8-10 – Specification of Qp_C as a function of qPi for ChromaArrayType equal to 1

qPi	< 30	30	31	32	33	34	35	36	37	38	39	40	41	42	43	> 43
Qp_C	= qPi	29	30	31	32	33	33	34	34	35	35	36	36	37	37	= qPi − 6

8.6.2 Scaling and transformation process

Inputs to this process are:

– a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,

– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding block,

– a variable cIdx specifying the colour component of the current block,

– a variable nTbS specifying the size of the current transform block.

Output of this process is the (nTbS)x(nTbS) array of residual samples r with elements r[x][y].

The quantization parameter qP is derived as follows:

– If cIdx is equal to 0, the following applies:

$$qP = \text{Clip3}(\ 0, 51 + \text{QpBdOffsetY}, \text{Qp}'\text{Y} + (\ \text{tu\_residual\_act\_flag}[\ \text{xTbY}\ ][\ \text{yTbY}\ ]\ ?\ \text{PpsActQpOffsetY} + \text{slice\_act\_y\_qp\_offset} : 0\ )\ ) \quad (8\text{-}291)$$

– Otherwise, if cIdx is equal to 1, the following applies:

$$qP = \text{Qp}'_{\text{Cb}} \quad (8\text{-}292)$$

– Otherwise (cIdx is equal to 2), the following applies:

$$qP = \text{Qp}'_{\text{Cr}} \quad (8\text{-}293)$$

The variables bitDepth, bdShift and tsShift are derived as follows:

$$\text{bitDepth} = (\ \text{cIdx} == 0\ )\ ?\ \text{BitDepth}_{\text{Y}} : \text{BitDepth}_{\text{C}} \quad (8\text{-}294)$$

$$\text{bdShift} = \text{Max}(\ 20 - \text{bitDepth}, \text{extended\_precision\_processing\_flag}\ ?\ 11 : 0\ ) \quad (8\text{-}295)$$

$$\text{tsShift} = 5 + \text{Log2}(\ \text{nTbS}\ ) \quad (8\text{-}296)$$

The variable rotateCoeffs is derived as follows:

– If all of the following conditions are true, rotateCoeffs is set equal to 1:

  – transform_skip_rotation_enabled_flag is equal to 1.

  – nTbS is equal to 4.

  – CuPredMode[xTbY][yTbY] is equal to MODE_INTRA.

– Otherwise, rotateCoeffs is set equal to 0.

The (nTbS)x(nTbS) array of residual samples r is derived as follows:

– If cu_transquant_bypass_flag is equal to 1, the following applies:

  – If rotateCoeffs is equal to 1, the residual sample array values r[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 are derived as follows:

$$r[\ x\ ][\ y\ ] = \text{TransCoeffLevel}[\ \text{xTbY}\ ][\ \text{yTbY}\ ][\ \text{cIdx}\ ][\ \text{nTbS} - x - 1\ ][\ \text{nTbS} - y - 1\ ] \quad (8\text{-}297)$$

  – Otherwise, the (nTbS)x(nTbS) array r is set equal to the (nTbS)x(nTbS) array of transform coefficients TransCoeffLevel[xTbY][yTbY][cIdx].

– Otherwise, the following ordered steps apply:

1. The scaling process for transform coefficients as specified in clause 8.6.3 is invoked with the transform block location (xTbY, yTbY), the size of the transform block nTbS, the colour component variable cIdx and the quantization parameter qP as inputs, and the output is an (nTbS)x(nTbS) array of scaled transform coefficients d.

2. The (nTbS)x(nTbS) array of residual samples r is derived as follows:

   – If transform_skip_flag[xTbY][yTbY][cIdx] is equal to 1, the residual sample array values r[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 are derived as follows:

$$r[\ x\ ][\ y\ ] = (\ \text{rotateCoeffs}\ ?\ d[\ \text{nTbS} - x - 1\ ][\ \text{nTbS} - y - 1\ ] : d[\ x\ ][\ y\ ]\ ) \ll \text{tsShift} \quad (8\text{-}298)$$

– Otherwise (transform_skip_flag[xTbY][yTbY][cIdx] is equal to 0), the transformation process for scaled transform coefficients as specified in clause 8.6.4 is invoked with the transform block location (xTbY, yTbY), the size of the transform block nTbS, the colour component variable cIdx and the (nTbS)x(nTbS) array of scaled transform coefficients d as inputs, and the output is an (nTbS)x(nTbS) array of residual samples r.

3. The residual sample values r[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 are modified as follows:

$$r[\ x\ ][\ y\ ] = (\ r[\ x\ ][\ y\ ] + (\ 1 \ll (\ bdShift - 1\ )\ )\ ) \gg bdShift \qquad (8\text{-}299)$$

8.6.3 Scaling process for transform coefficients

Inputs to this process are:

– a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,

– a variable nTbS specifying the size of the current transform block,

– a variable cIdx specifying the colour component of the current block,

– a variable qP specifying the quantization parameter.

Output of this process is the (nTbS)x(nTbS) array d of scaled transform coefficients with elements d[x][y].

The variables log2TransformRange, bdShift, coeffMin and coeffMax are derived as follows:

– If cIdx is equal to 0, the following applies:

$$\text{log2TransformRange} = \text{extended\_precision\_processing\_flag} \ ?\ \text{Max}(\ 15, \text{BitDepth}_Y + 6\ ) : 15 \qquad (8\text{-}300)$$

$$\text{bdShift} = \text{BitDepth}_Y + \text{Log2}(\ \text{nTbS}\ ) + 10 - \text{log2TransformRange} \qquad (8\text{-}301)$$

$$\text{coeffMin} = \text{CoeffMin}_Y \qquad (8\text{-}302)$$

$$\text{coeffMax} = \text{CoeffMax}_Y \qquad (8\text{-}303)$$

– Otherwise, the following applies:

$$\text{log2TransformRange} = \text{extended\_precision\_processing\_flag} \ ?\ \text{Max}(\ 15, \text{BitDepth}_C + 6\ ) : 15 \qquad (8\text{-}304)$$

$$\text{bdShift} = \text{BitDepth}_C + \text{Log2}(\ \text{nTbS}\ ) + 10 - \text{log2TransformRange} \qquad (8\text{-}305)$$

$$\text{coeffMin} = \text{CoeffMin}_C \qquad (8\text{-}306)$$

$$\text{coeffMax} = \text{CoeffMax}_C \qquad (8\text{-}307)$$

The list levelScale[ ] is specified as levelScale[k] = { 40, 45, 51, 57, 64, 72 } with k = 0..5.

For the derivation of the scaled transform coefficients d[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1, the following applies:

– The scaling factor m[x][y] is derived as follows:

  – If one or more of the following conditions are true, m[x][y] is set equal to 16:

    – scaling_list_enabled_flag is equal to 0.

    – transform_skip_flag[xTbY][yTbY] is equal to 1 and nTbS is greater than 4.

  – Otherwise, the following applies:

$$m[\ x\ ][\ y\ ] = \text{ScalingFactor}[\ \text{sizeId}\ ][\ \text{matrixId}\ ][\ x\ ][\ y\ ] \qquad (8\text{-}308)$$

Where sizeId is specified in Table 7-3 for the size of the quantization matrix equal to (nTbS)x(nTbS) and matrixId is specified in Table 7-4 for sizeId, CuPredMode[xTbY][yTbY] and cIdx, respectively.

– The scaled transform coefficient d[x][y] is derived as follows:

d[x][y] =
Clip3(coeffMin, coeffMax, ((TransCoeffLevel[xTbY][yTbY][cIdx][x][y] * m[x][y] *
levelScale[qP % 6] << (qP / 6)) + (1 << (bdShift − 1))) >> bdShift) (8-309)

8.6.4 Transformation process for scaled transform coefficients

8.6.4.1 General

Inputs to this process are:

– a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,

– a variable nTbS specifying the size of the current transform block,

– a variable cIdx specifying the colour component of the current block,

– an (nTbS)x(nTbS) array d of scaled transform coefficients with elements d[x][y].

Output of this process is the (nTbS)x(nTbS) array r of residual samples with elements r[x][y].

The variables coeffMin and coeffMax are derived as follows:

– If cIdx is equal to 0, the following applies:

$$\text{coeffMin} = \text{CoeffMin}_{\text{Y}} \tag{8-310}$$

$$\text{coeffMax} = \text{CoeffMax}_{\text{Y}} \tag{8-311}$$

– Otherwise, the following applies:

$$\text{coeffMin} = \text{CoeffMin}_{\text{C}} \tag{8-312}$$

$$\text{coeffMax} = \text{CoeffMax}_{\text{C}} \tag{8-313}$$

Depending on the values of CuPredMode[xTbY][yTbY], nTbS and cIdx, the variable trType is derived as follows:

– If CuPredMode[xTbY][yTbY] is equal to MODE_INTRA, nTbS is equal to 4 and cIdx is equal to 0, trType is set equal to 1.

– Otherwise, trType is set equal to 0.

The (nTbS)x(nTbS) array r of residual samples is derived by the following ordered steps:

1. Each (vertical) column of scaled transform coefficients d[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 is transformed to e[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 by invoking the one-dimensional transformation process as specified in clause 8.6.4.2 for each column x = 0..nTbS − 1 with the size of the transform block nTbS, the list d[x][y] with y = 0..nTbS − 1 and the transform type variable trType as inputs, and the output is the list e[x][y] with y = 0..nTbS − 1.

2. The intermediate sample values g[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 are derived as follows:

g[x][y] = Clip3(coeffMin, coeffMax, (e[x][y] + 64) >> 7) (8-314)

3. Each (horizontal) row of the resulting array g[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 is transformed to r[x][y] with x = 0..nTbS − 1, y = 0..nTbS − 1 by invoking the one-dimensional transformation process as specified in clause 8.6.4.2 for each row y = 0..nTbS − 1 with the size of the transform block nTbS, the list g[x][y] with x = 0..nTbS − 1 and the transform type variable trType as inputs, and the output is the list r[x][y] with x = 0..nTbS − 1.

8.6.4.2 Transformation process

Inputs to this process are:

– a variable nTbS specifying the sample size of scaled transform coefficients,

– a list of scaled transform coefficients x with elements x[j], with j = 0..nTbS − 1.

– a transform type variable trType

Output of this process is the list of transformed samples y with elements y[i], with i = 0..nTbS − 1.

Depending on the value of trType, the following applies:

– If trType is equal to 1, the following transform matrix multiplication applies:

$$y[\,i\,] = \sum_{j=0}^{nTbS-1} transMatrix[i][\,j\,] * x[\,j\,] \text{ with i = 0..nTbS − 1} \qquad (8\text{-}315)$$

where the transform coefficient array transMatrix is specified as follows:

transMatrix = (8-316)

```
{
{29  55  74  84}
{74  74   0 -74}
{84 -29 -74  55}
{55 -84  74 -29}
}
```

– Otherwise (trType is equal to 0), the following transform matrix multiplication applies:

$$y[\,i\,] = \sum_{j=0}^{nTbS-1} transMatrix[i][\,j * 2^{5-Log2(nTbS)}\,] * x[\,j\,] \text{ with i = 0..nTbS − 1,} \qquad (8\text{-}317)$$

where the transform coefficient array transMatrix is specified as follows:

transMatrix[m][n] = transMatrixCol0to15[m][n] with m = 0..15, n = 0..31 (8-318)

transMatrixCol0to15 = (8-319)

```
{
{64  64  64  64  64  64  64  64  64  64  64  64  64  64  64  64}
{90  90  88  85  82  78  73  67  61  54  46  38  31  22  13   4}
{90  87  80  70  57  43  25   9  -9 -25 -43 -57 -70 -80 -87 -90}
{90  82  67  46  22  -4 -31 -54 -73 -85 -90 -88 -78 -61 -38 -13}
{89  75  50  18 -18 -50 -75 -89 -89 -75 -50 -18  18  50  75  89}
{88  67  31 -13 -54 -82 -90 -78 -46  -4  38  73  90  85  61  22}
{87  57   9 -43 -80 -90 -70 -25  25  70  90  80  43  -9 -57 -87}
{85  46 -13 -67 -90 -73 -22  38  82  88  54  -4 -61 -90 -78 -31}
{83  36 -36 -83 -83 -36  36  83  83  36 -36 -83 -83 -36  36  83}
{82  22 -54 -90 -61  13  78  85  31 -46 -90 -67   4  73  88  38}
{80   9 -70 -87 -25  57  90  43 -43 -90 -57  25  87  70  -9 -80}
{78  -4 -82 -73  13  85  67 -22 -88 -61  31  90  54 -38 -90 -46}
{75 -18 -89 -50  50  89  18 -75 -75  18  89  50 -50 -89 -18  75}
{73 -31 -90 -22  78  67 -38 -90 -13  82  61 -46 -88  -4  85  54}
{70 -43 -87   9  90  25 -80 -57  57  80 -25 -90  -9  87  43 -70}
{67 -54 -78  38  85 -22 -90   4  90  13 -88 -31  82  46 -73 -61}
{64 -64 -64  64  64 -64 -64  64  64 -64 -64  64  64 -64 -64  64}
{61 -73 -46  82  31 -88 -13  90  -4 -90  22  85 -38 -78  54  67}
{57 -80 -25  90  -9 -87  43  70 -70 -43  87   9 -90  25  80 -57}
{54 -85  -4  88 -46 -61  82  13 -90  38  67 -78 -22  90 -31 -73}
{50 -89  18  75 -75 -18  89 -50 -50  89 -18 -75  75  18 -89  50}
{46 -90  38  54 -90  31  61 -88  22  67 -85  13  73 -82   4  78}
{43 -90  57  25 -87  70   9 -80  80  -9 -70  87 -25 -57  90 -43}
{38 -88  73  -4 -67  90 -46 -31  85 -78  13  61 -90  54  22 -82}
{36 -83  83 -36 -36  83 -83  36  36 -83  83 -36 -36  83 -83  36}
{31 -78  90 -61   4  54 -88  82 -38 -22  73 -90  67 -13 -46  85}
{25 -70  90 -80  43   9 -57  87 -87  57  -9 -43  80 -90  70 -25}
{22 -61  85 -90  73 -38  -4  46 -78  90 -82  54 -13 -31  67 -88}
{18 -50  75 -89  89 -75  50 -18 -18  50 -75  89 -89  75 -50  18}
{13 -38  61 -78  88 -90  85 -73  54 -31   4  22 -46  67 -82  90}
{ 9 -25  43 -57  70 -80  87 -90  90 -87  80 -70  57 -43  25  -9}
{ 4 -13  22 -31  38 -46  54 -61  67 -73  78 -82  85 -88  90 -90}
},
```

transMatrix[m][n] = transMatrixCol16to31[m − 16][n] with m = 16..31, n = 0..31, (8-320)

transMatrixCol16to31 = (8-321)

```
{
{ 64  64  64  64  64  64  64  64  64  64  64  64  64  64  64  64}
{ -4 -13 -22 -31 -38 -46 -54 -61 -67 -73 -78 -82 -85 -88 -90 -90}
{-90 -87 -80 -70 -57 -43 -25  -9   9  25  43  57  70  80  87  90}
{ 13  38  61  78  88  90  85  73  54  31   4 -22 -46 -67 -82 -90}
{ 89  75  50  18 -18 -50 -75 -89 -89 -75 -50 -18  18  50  75  89}
{-22 -61 -85 -90 -73 -38   4  46  78  90  82  54  13 -31 -67 -88}
{-87 -57  -9  43  80  90  70  25 -25 -70 -90 -80 -43   9  57  87}
{ 31  78  90  61   4 -54 -88 -82 -38  22  73  90  67  13 -46 -85}
{ 83  36 -36 -83 -83 -36  36  83  83  36 -36 -83 -83 -36  36  83}
{-38 -88 -73  -4  67  90  46 -31 -85 -78 -13  61  90  54 -22 -82}
{-80  -9  70  87  25 -57 -90 -43  43  90  57 -25 -87 -70   9  80}
{ 46  90  38 -54 -90 -31  61  88  22 -67 -85 -13  73  82   4 -78}
{ 75 -18 -89 -50  50  89  18 -75 -75  18  89  50 -50 -89 -18  75}
{-54 -85   4  88  46 -61 -82  13  90  38 -67 -78  22  90  31 -73}
{-70  43  87  -9 -90 -25  80  57 -57 -80  25  90   9 -87 -43  70}
{ 61  73 -46 -82  31  88 -13 -90  -4  90  22 -85 -38  78  54 -67}
{ 64 -64 -64  64  64 -64 -64  64  64 -64 -64  64  64 -64 -64  64}
{-67 -54  78  38 -85 -22  90   4 -90  13  88 -31 -82  46  73 -61}
{-57  80  25 -90   9  87 -43 -70  70  43 -87  -9  90 -25 -80  57}
{ 73  31 -90  22  78 -67 -38  90 -13 -82  61  46 -88   4  85 -54}
{ 50 -89  18  75 -75 -18  89 -50 -50  89 -18 -75  75  18 -89  50}
{-78  -4  82 -73 -13  85 -67 -22  88 -61 -31  90 -54 -38  90 -46}
{-43  90 -57 -25  87 -70  -9  80 -80   9  70 -87  25  57 -90  43}
{ 82 -22 -54  90 -61 -13  78 -85  31  46 -90  67   4 -73  88 -38}
{ 36 -83  83 -36 -36  83 -83  36  36 -83  83 -36 -36  83 -83  36}
{-85  46  13 -67  90 -73  22  38 -82  88 -54  -4  61 -90  78 -31}
{-25  70 -90  80 -43  -9  57 -87  87 -57   9  43 -80  90 -70  25}
{ 88 -67  31  13 -54  82 -90  78 -46   4  38 -73  90 -85  61 -22}
{ 18 -50  75 -89  89 -75  50 -18 -18  50 -75  89 -89  75 -50  18}
{-90  82 -67  46 -22  -4  31 -54  73 -85  90 -88  78 -61  38 -13}
{ -9  25 -43  57 -70  80 -87  90 -90  87 -80  70 -57  43 -25   9}
{ 90 -90  88 -85  82 -78  73 -67  61 -54  46 -38  31 -22  13  -4}
}
```

8.6.5 Residual modification process for blocks using a transform bypass

Inputs to this process are:

– a variable mDir specifying the residual modification direction,

– a variable nTbS specifying the transform block size,

– an (nTbS)x(nTbS) array of residual samples r with elements r[x][y].

Output of this process is the modified (nTbS)x(nTbS) array of residual samples.

Depending upon the value of mDir, the (nTbS)x(nTbS) array of samples r is modified as follows:

– If mDir is equal to 0 (horizontal direction), the array values r[x][y] are modified as follows, for x proceeding over the values 1..nTbS − 1 and y = 0..nTbS − 1:

r[x][y] += r[x − 1][y] (8-322)

– Otherwise (vertical direction), the array values r[x][y] are modified as follows, for y proceeding over the values 1..nTbS − 1 and for x = 0..nTbS − 1:

r[x][y] += r[x][y − 1] (8-323)

8.6.6 Residual modification process for transform blocks using cross-component prediction

This process is only invoked when ChromaArrayType is equal to 3.

Inputs to this process are:

– a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,

– a variable nTbS specifying the transform block size,

– a variable cIdx specifying the colour component of the current block,
– an (nTbS)x(nTbS) array of luma residual samples r_Y with elements r_Y[x][y],
– an (nTbS)x(nTbS) array of residual samples r with elements r[x][y].

Output of this process is the modified (nTbS)x(nTbS) array r of residual samples.

The (nTbS)x(nTbS) array of residual samples r with x = 0..nTbS − 1, y = 0..nTbS − 1 is modified as follows:

$$r[\ x\ ][\ y\ ] \mathrel{+}= (\ \mathrm{ResScaleVal}[\ \mathrm{cIdx}\ ][\ \mathrm{xTbY}\ ][\ \mathrm{yTbY}\ ] * ((\ r_Y[\ x\ ][\ y\ ] \ll \mathrm{BitDepth}_C\ ) \gg \mathrm{BitDepth}_Y\ )\ ) \gg 3 \quad (8\text{-}324)$$

It is a requirement of bitstream conformance that the luma residual samples rY[x][y] shall be in the range of CoeffMinY to CoeffMaxY, inclusive, and the input residual samples r[x][y] shall be in the range of CoeffMinC to CoeffMaxC, inclusive.

8.6.7 Picture construction process prior to in-loop filter process

Inputs to this process are:

– a location (xCurr, yCurr) specifying the top-left sample of the current block relative to the top-left sample of the current picture component,
– the variables nCurrSw and nCurrSh specifying the width and height, respectively, of the current block,
– a variable cIdx specifying the colour component of the current block,
– an (nCurrSw)x(nCurrSh) array predSamples specifying the predicted samples of the current block,
– an (nCurrSw)x(nCurrSh) array resSamples specifying the residual samples of the current block.

Depending on the value of the colour component cIdx, the following assignments are made:

– If cIdx is equal to 0, recSamples corresponds to the reconstructed picture sample array S_L and the function clipCidx1 corresponds to $\mathrm{Clip1}_Y$.
– Otherwise, if cIdx is equal to 1, recSamples corresponds to the reconstructed chroma sample array S_{Cb} and the function clipCidx1 corresponds to $\mathrm{Clip1}_C$.
– Otherwise (cIdx is equal to 2), recSamples corresponds to the reconstructed chroma sample array S_{Cr} and the function clipCidx1 corresponds to $\mathrm{Clip1}_C$.

The (nCurrSw)x(nCurrSh) block of the reconstructed sample array recSamples at location (xCurr, yCurr) is derived as follows:

$$\mathrm{recSamples}[\ \mathrm{xCurr} + i\ ][\ \mathrm{yCurr} + j\ ] = \mathrm{clipCidx1}(\ \mathrm{predSamples}[\ i\ ][\ j\ ] + \mathrm{resSamples}[\ i\ ][\ j\ ]\ ) \quad (8\text{-}325)$$
$$\text{with } i = 0..\mathrm{nCurrSw} - 1,\ j = 0..\mathrm{nCurrSh} - 1$$

8.6.8 Residual modification process for blocks using adaptive colour transform

This process is only invoked when ChromaArrayType is equal to 3.

8.6.8.1 General

Inputs to this process are:

– a sample location (xTb0, yTb0) specifying the top-left sample of the current block relative to the top left sample of the current picture,
– a variable log2TrafoSize specifying the size of the current block,
– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,
– an array resSampleArrayL specifying the luma residual samples of the current block,
– an array resSampleArrayCb specifying the Cb residual samples of the current block,
– an array resSampleArrayCr specifying the Cr residual samples of the current block.

Outputs of this process are the modified arrays resSampleY, resSampleCb and resSampleCr of residual samples of the current block.

Depending on the value of split_transform_flag[xTb0][yTb0][trafoDepth], the following applies:

– If split_transform_flag[xTb0][yTb0][trafoDepth] is equal to 1, the following ordered steps apply:

1. The variables xTb1 and yTb1 are derived as follows:
   – The variable xTb1 is set equal to xTb0 + (1 << (log2TrafoSize − 1)).
   – The variable yTb1 is set equal to yTb0 + (1 << (log2TrafoSize − 1)).
2. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (xTb0, yTb0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the residual sample arrays resSampeArrayL, resSampleArrayCb and resSampleArrayCr as inputs and the output are modified versions of resSampeArrayL, resSampleArrayCb and resSampleArrayCr.
3. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (xTb1, yTb0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the residual sample arrays resSampeArrayL, resSampleArrayCb and resSampleArrayCr as inputs and the output are modified versions of resSampeArrayL, resSampleArrayCb and resSampleArrayCr.
4. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (xTb0, yTb1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the residual sample arrays resSampeArrayL, resSampleArrayCb and resSampleArrayCr as inputs and the output are modified versions of resSampeArrayL, resSampleArrayCb and resSampleArrayCr.
5. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (xTb1, yTb1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the residual sample arrays resSampeArrayL, resSampleArrayCb and resSampleArrayCr as inputs and the output are modified versions of resSampeArrayL, resSampleArrayCb and resSampleArrayCr.

– Otherwise (split_transform_flag[xTb0][yTb0][trafoDepth] is equal to 0), the following ordered steps apply:

1. The variable nTbS specifying the current transform block is set equal to (1 << log2TrafoSize).
2. The variable nCbS specifying the size of the current coding block is derived as nCbS = 1 << (log2TrafoSize + trafDepth).
3. The sample location (xTbInCb, yTbInCb) specifying the top-left sample of the current transform block relative to the top-left sample of the current coding block are derived as xTbInCb = xTb0 & (nCbS − 1) and yTbInCb = yTb0 & (nCbS − 1).

When tu_residual_act_flag[xTb0][yTb0] is equal to 1, the adaptive colour transformation process as specified in clause 8.6.8.2 is invoked with the sample location (xTb0, yTb0) set to (xTbInCb, yTbInCb), the variable blkSize set equal to nTbS, the array rY set equal to resSampleArrayL, the array rCb set equal to resSampleArrayCb, and the array rCr set equal to resSampleArrayCr as inputs, and the outputs are modified versions of the three residual sample arrays.

8.6.8.2 Adaptive colour transformation process

Inputs to this process are:

– a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top left sample of the current coding block,
– a variable blkSize specifying the block size of the transform block to be modified,
– an array of luma residual samples rY of the current coding block,
– an array of chroma residual samples rCb of the current coding block,
– an array of chroma residual samples rCr of the current coding block.

Outputs of this process are:

– a modified array rY of luma residual samples,

– a modified array rCb of chroma residual samples,

– a modified array rCr of chroma residual samples.

The arrays of residual samples rY, rCb and rCr with x = xTb0..xTb0 + blkSize − 1, y = yTb0..yTb0 + blkSize − 1 are modified as follows:

$$rY[\ x\ ][\ y\ ] = Clip3(\ CoeffMinY, CoeffMaxY, rY[\ x\ ][\ y\ ]\ ) \quad (8\text{-}326)$$

$$rCb[\ x\ ][\ y\ ] = Clip3(\ CoeffMinC, CoeffMaxC, rCb[\ x\ ][\ y\ ]\ ) \quad (8\text{-}327)$$

$$rCr[\ x\ ][\ y\ ] = Clip3(\ CoeffMinC, CoeffMaxC, rCr[\ x\ ][\ y\ ]\ ) \quad (8\text{-}328)$$

– When cu_transquant_bypass_flag is equal to 0, the following ordered steps apply:

1. The variables deltaBDY and deltaBDC are derived as follows:

$$deltaBDY = Max(\ BitDepth_Y, BitDepth_C\ ) - BitDepth_Y \quad (8\text{-}329)$$

$$deltaBDC = Max(\ BitDepth_Y, BitDepth_C\ ) - BitDepth_C \quad (8\text{-}330)$$

$$offsetBDY = deltaBDY\ ?\ 1 << (\ deltaBDY - 1\ ) : 0 \quad (8\text{-}331)$$

$$offsetBDC = deltaBDC\ ?\ 1 << (\ deltaBDC - 1\ ) : 0 \quad (8\text{-}332)$$

2. Residual samples rY[x][y], rCb[x][y] and rCr[x][y] are modified as follows:

$$rY[\ x\ ][\ y\ ] = rY[\ x\ ][\ y\ ] << deltaBDY \quad (8\text{-}333)$$

$$rCb[\ x\ ][\ y\ ] = rCb[\ x\ ][\ y\ ] << (\ deltaBDC + 1\ ) \quad (8\text{-}334)$$

$$rCr[\ x\ ][\ y\ ] = rCr[\ x\ ][\ y\ ] << (\ deltaBDC + 1\ ) \quad (8\text{-}335)$$

– Residual samples rY[x][y], rCb[x][y] and rCr[x][y] are modified as follows:

$$tmp = rY[\ x\ ][\ y\ ] - (\ rCb[\ x\ ][\ y\ ] >> 1\ ) \quad (8\text{-}336)$$

$$rY[\ x\ ][\ y\ ] = tmp + rCb[\ x\ ][\ y\ ] \quad (8\text{-}337)$$

$$rCb[\ x\ ][\ y\ ] = tmp - (\ rCr[\ x\ ][\ y\ ] >> 1\ ) \quad (8\text{-}338)$$

$$rCr[\ x\ ][\ y\ ] \mathrel{+}= rCb[\ x\ ][\ y\ ] \quad (8\text{-}339)$$

– When cu_transquant_bypass_flag is equal to 0, the following applies:

$$rY[\ x\ ][\ y\ ] = (\ rY[\ x\ ][\ y\ ] + offsetBDY\ ) >> deltaBDY \quad (8\text{-}340)$$

$$rCb[\ x\ ][\ y\ ] = (\ rCb[\ x\ ][\ y\ ] + offsetBDC\ ) >> deltaBDC \quad (8\text{-}341)$$

$$rCr[\ x\ ][\ y\ ] = (\ rCr[\ x\ ][\ y\ ] + offsetBDC\ ) >> deltaBDC \quad (8\text{-}342)$$

8.7 In-loop filter process

8.7.1 General

This clause specifies the application of two in-loop filters. When the in-loop filter process is specified as optional in Annex A, the application of either or both of these filters is optional.

The two in-loop filters, namely deblocking filter and sample adaptive offset filter, are applied as specified by the following ordered steps:

1. For the deblocking filter, the following applies:

– The deblocking filter process as specified in clause 8.7.2 is invoked with the reconstructed picture sample array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} as inputs, and the modified reconstructed picture sample array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} after deblocking as outputs.

– The array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} are assigned to the array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} (which represent the decoded picture), respectively.

2. When sample_adaptive_offset_enabled_flag is equal to 1, the following applies:

– The sample adaptive offset process as specified in clause 8.7.3 is invoked with the reconstructed picture sample array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} as inputs, and the modified reconstructed picture sample array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} after sample adaptive offset as outputs.

– The array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} are assigned to the array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} (which represent the decoded picture), respectively.

8.7.2 Deblocking filter process

8.7.2.1 General

Inputs to this process are the reconstructed picture prior to deblocking, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

Outputs of this process are the modified reconstructed picture after deblocking, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

The vertical edges in a picture are filtered first. Then the horizontal edges in a picture are filtered with samples modified by the vertical edge filtering process as input. The vertical and horizontal edges in the CTBs of each CTU are processed separately on a coding unit basis. The vertical edges of the coding blocks in a coding unit are filtered starting with the edge on the left-hand side of the coding blocks proceeding through the edges towards the right-hand side of the coding blocks in their geometrical order. The horizontal edges of the coding blocks in a coding unit are filtered starting with the edge on the top of the coding blocks proceeding through the edges towards the bottom of the coding blocks in their geometrical order.

NOTE – Although the filtering process is specified on a picture basis in this Specification, the filtering process can be implemented on a coding unit basis with an equivalent result, provided the decoder properly accounts for the processing dependency order so as to produce the same output values.

The deblocking filter process is applied to all prediction block edges and transform block edges of a picture, except the following types of edges:

– Edges that are at the boundary of the picture,
– Edges that coincide with tile boundaries when loop_filter_across_tiles_enabled_flag is equal to 0,
– Edges that coincide with upper or left boundaries of slices with slice_loop_filter_across_slices_enabled_flag equal to 0 or slice_deblocking_filter_disabled_flag equal to 1,
– Edges within slices with slice_deblocking_filter_disabled_flag equal to 1,
– Edges that do not correspond to 8x8 sample grid boundaries of the considered component,
– Edges within chroma components for which both sides of the edge use inter prediction,
– Edges of chroma transform blocks that are not edges of the associated transform unit.

The edge type, vertical or horizontal, is represented by the variable edgeType as specified in Table 8-11.

Table 8-11 – Name of association to edgeType

edgeType	Name of edgeType
0 (vertical edge)	EDGE_VER
1 (horizontal edge)	EDGE_HOR

When slice_deblocking_filter_disabled_flag of the current slice is equal to 0, for each coding unit with luma coding block size log2CbSize and location of top-left sample of the luma coding block (xCb, yCb), the vertical edges are filtered by the following ordered steps:

1. The luma coding block size nCbS is set equal to 1 << log2CbSize.
2. The variable filterLeftCbEdgeFlag is derived as follows:
   – If one or more of the following conditions are true, filterLeftCbEdgeFlag is set equal to 0:
     – The left boundary of the current luma coding block is the left boundary of the picture.
     – The left boundary of the current luma coding block is the left boundary of the tile and loop_filter_across_tiles_enabled_flag is equal to 0.
     – The left boundary of the current luma coding block is the left boundary of the slice and slice_loop_filter_across_slices_enabled_flag is equal to 0.
   – Otherwise, filterLeftCbEdgeFlag is set equal to 1.
3. All elements of the two-dimensional (nCbS)x(nCbS) array verEdgeFlags are initialized to be equal to zero.
4. The derivation process of transform block boundary specified in clause 8.7.2.2 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the transform block size log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable filterLeftCbEdgeFlag, the array verEdgeFlags and the variable edgeType set equal to EDGE_VER as inputs, and the modified array verEdgeFlags as output.
5. The derivation process of prediction block boundary specified in clause 8.7.2.3 is invoked with the luma coding block size log2CbSize, the prediction partition mode PartMode, the array verEdgeFlags and the variable edgeType set equal to EDGE_VER as inputs, and the modified array verEdgeFlags as output.
6. The derivation process of the boundary filtering strength specified in clause 8.7.2.4 is invoked with the reconstructed luma picture sample array prior to deblocking $recPicture_L$, the luma location (xCb, yCb), the luma coding block size log2CbSize, the variable edgeType set equal to EDGE_VER and the array verEdgeFlags as inputs, and an (nCbS)x(nCbS) array verBs as output.
7. The vertical edge filtering process for a coding unit as specified in clause 8.7.2.5.1 is invoked with the reconstructed picture prior to deblocking, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$, the luma location (xCb, yCb), the luma coding block size log2CbSize and the array verBs as inputs, and the modified reconstructed picture, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$, as output.

When slice_deblocking_filter_disabled_flag of the current slice is equal to 0, for each coding unit with luma coding block size log2CbSize and location of top-left sample of the luma coding block (xCb, yCb), the horizontal edges are filtered by the following ordered steps:

1. The luma coding block size nCbS is set equal to 1 << log2CbSize.
2. The variable filterTopCbEdgeFlag is derived as follows:
   – If one or more of the following conditions are true, the variable filterTopCbEdgeFlag is set equal to 0:
     – The top boundary of the current luma coding block is the top boundary of the picture.
     – The top boundary of the current luma coding block is the top boundary of the tile and loop_filter_across_tiles_enabled_flag is equal to 0.
     – The top boundary of the current luma coding block is the top boundary of the slice and slice_loop_filter_across_slices_enabled_flag is equal to 0.
   – Otherwise, the variable filterTopCbEdgeFlag is set equal to 1.
3. All elements of the two-dimensional (nCbS)x(nCbS) array horEdgeFlags are initialized to zero.
4. The derivation process of transform block boundary specified in clause 8.7.2.2 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the transform block size log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable filterTopCbEdgeFlag, the array horEdgeFlags and the variable edgeType set equal to EDGE_HOR as inputs, and the modified array horEdgeFlags as output.
5. The derivation process of prediction block boundary specified in clause 8.7.2.3 is invoked with the luma coding block size log2CbSize, the prediction partition mode PartMode, the array horEdgeFlags and the variable edgeType set equal to EDGE_HOR as inputs, and the modified array horEdgeFlags as output.
6. The derivation process of the boundary filtering strength specified in clause 8.7.2.4 is invoked with the reconstructed luma picture sample array prior to deblocking $recPicture_L$, the luma location (xCb, yCb), the luma coding block size log2CbSize, the variable edgeType set equal to EDGE_HOR and the array horEdgeFlags as inputs, and an (nCbS)x(nCbS) array horBs as output.

7. The horizontal edge filtering process for a coding unit as specified in clause 8.7.2.5.2 is invoked with the modified reconstructed picture, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$, the luma location (xCb, yCb), the luma coding block size log2CbSize, and the array horBs as inputs and the modified reconstructed picture, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$, as output.

8.7.2.2 Derivation process of transform block boundary

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xB0, yB0) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,

– a variable log2TrafoSize specifying the size of the current block,

– a variable trafoDepth,

– a variable filterEdgeFlag,

– a two-dimensional (nCbS)x(nCbS) array edgeFlags,

– a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered.

Output of this process is the modified two-dimensional (nCbS)x(nCbS) array edgeFlags.

Depending on the value of split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth], the following applies:

– If split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth] is equal to 1, the following ordered steps apply:

1. The variables xB1 and yB1 are derived as follows:

   – The variable xB1 is set equal to xB0 + (1 << (log2TrafoSize − 1)).

   – The variable yB1 is set equal to yB0 + (1 << (log2TrafoSize − 1)).

2. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable filterEdgeFlag, the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags.

3. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB1, yB0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable filterEdgeFlag, the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags.

4. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB0, yB1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable filterEdgeFlag, the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags.

5. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (xCb, yCb), the luma location (xB1, yB1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable filterEdgeFlag, the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags.

– Otherwise (split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth] is equal to 0), the following applies:

  – If edgeType is equal to EDGE_VER, the value of edgeFlags[xB0][yB0 + k] for k = 0..(1 << log2TrafoSize) − 1 is derived as follows:

    – If xB0 is equal to 0, edgeFlags[xB0][yB0 + k] is set equal to filterEdgeFlag.

    – Otherwise, edgeFlags[xB0][yB0 + k] is set equal to 1.

  – Otherwise (edgeType is equal to EDGE_HOR), the value of edgeFlags[xB0 + k][yB0] for k = 0..(1 << log2TrafoSize) − 1 is derived as follows:

    – If yB0 is equal to 0, edgeFlags[xB0 + k][yB0] is set equal to filterEdgeFlag.

    – Otherwise, edgeFlags[xB0 + k][yB0] is set equal to 1.

8.7.2.3 Derivation process of prediction block boundary

Inputs to this process are:

– a variable log2CbSize specifying the luma coding block size,

– a prediction partition mode PartMode,

– a two-dimensional (nCbS)x(nCbS) array edgeFlags,

– a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered.

Output of this process is the modified two-dimensional (nCbS)x(nCbS) array edgeFlags.

Depending on the values of edgeType and PartMode, the following applies for k = 0..(1 << log2CbSize) − 1:

– If edgeType is equal to EDGE_VER, the following applies:
  – When PartMode is equal to PART_Nx2N or PART_NxN, edgeFlags[1 << (log2CbSize − 1)][k] is set equal to 1.
  – When PartMode is equal to PART_nLx2N, edgeFlags[1 << (log2CbSize − 2)][k] is set equal to 1.
  – When PartMode is equal to PART_nRx2N, edgeFlags[3 * (1 << (log2CbSize − 2))][k] is set equal to 1.
– Otherwise (edgeType is equal to EDGE_HOR), the following applies:
  – When PartMode is equal to PART_2NxN or PART_NxN, edgeFlags[k][1 << (log2CbSize − 1)] is set equal to 1.
  – When PartMode is equal to PART_2NxnU, edgeFlags[k][1 << (log2CbSize − 2)] is set equal to 1.
  – When PartMode is equal to PART_2NxnD, edgeFlags[k][3 * (1 << (log2CbSize − 2))] is set equal to 1.

8.7.2.4 Derivation process of boundary filtering strength

Inputs to this process are:

– a luma picture sample array recPicture_L,

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block,

– a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,

– a two-dimensional (nCbS)x(nCbS) array edgeFlags.

Output of this process is a two-dimensional (nCbS)x(nCbS) array bS specifying the boundary filtering strength.

The variables xD_i, yD_j, xN and yN are derived as follows:

– If edgeType is equal to EDGE_VER, xD_i is set equal to (i << 3), yD_j is set equal to (j << 2), xN is set equal to (1 << (log2CbSize − 3)) − 1 and yN is set equal to (1 << (log2CbSize − 2)) − 1.

– Otherwise (edgeType is equal to EDGE_HOR), xD_i is set equal to (i << 2), yD_j is set equal to (j << 3), xN is set equal to (1 << (log2CbSize − 2)) − 1 and yN is set equal to (1 << (log2CbSize − 3)) − 1.

For xD_i with i = 0..xN and yD_j with j = 0..yN, the following applies:

– If edgeFlags[xD_i][yD_j] is equal to 0, the variable bS[xD_i][yD_j] is set equal to 0.
– Otherwise (edgeFlags[xD_i][yD_j] is equal to 1), the following applies:
  – The sample values p_0 and q_0 are derived as follows:
    – If edgeType is equal to EDGE_VER, p_0 is set equal to recPicture_L[xCb + xD_i − 1][yCb + yD_j] and q_0 is set equal to recPicture_L[xCb + xD_i][yCb + yD_j].
    – Otherwise (edgeType is equal to EDGE_HOR), p_0 is set equal to recPicture_L[xCb + xD_i][yCb + yD_j − 1] and q_0 is set equal to recPicture_L[xCb + xD_i][yCb + yD_j].
  – The variable bS[xD_i][yD_j] is derived as follows:
    – If the sample p_0 or q_0 is in the luma coding block of a coding unit coded with intra prediction mode, bS[xD_i][yD_j] is set equal to 2.

– Otherwise, if the block edge is also a transform block edge and the sample p_0 or q_0 is in a luma transform block which contains one or more non-zero transform coefficient levels, bS[xD_i][yD_j] is set equal to 1.

– Otherwise, if one or more of the following conditions are true, bS[xD_i][yD_j] is set equal to 1:

  – For the prediction of the luma prediction block containing the sample p_0 different reference pictures or a different number of motion vectors are used than for the prediction of the luma prediction block containing the sample q_0.

    NOTE 1 – The determination of whether the reference pictures used for the two luma prediction blocks are the same or different is based only on which pictures are referenced, without regard to whether a prediction is formed using an index into reference picture list 0 or an index into reference picture list 1, and also without regard to whether the index position within a reference picture list is different.

    NOTE 2 – The number of motion vectors that are used for the prediction of a luma prediction block with top-left luma sample covering (xPb, yPb), is equal to PredFlagL0[xPb][yPb] + PredFlagL1[xPb][yPb].

  – One motion vector is used to predict the luma prediction block containing the sample p_0 and one motion vector is used to predict the luma prediction block containing the sample q_0, and the absolute difference between the horizontal or vertical component of the motion vectors used is greater than or equal to 4 in units of quarter luma samples.

  – Two motion vectors and two different reference pictures are used to predict the luma prediction block containing the sample p_0, two motion vectors for the same two reference pictures are used to predict the luma prediction block containing the sample q_0 and the absolute difference between the horizontal or vertical component of the two motion vectors used in the prediction of the two luma prediction blocks for the same reference picture is greater than or equal to 4 in units of quarter luma samples.

  – Two motion vectors for the same reference picture are used to predict the luma prediction block containing the sample p_0, two motion vectors for the same reference picture are used to predict the luma prediction block containing the sample q_0 and both of the following conditions are true:

    – The absolute difference between the horizontal or vertical component of list 0 motion vectors used in the prediction of the two luma prediction blocks is greater than or equal to 4 in quarter luma samples, or the absolute difference between the horizontal or vertical component of the list 1 motion vectors used in the prediction of the two luma prediction blocks is greater than or equal to 4 in units of quarter luma samples.

    – The absolute difference between the horizontal or vertical component of list 0 motion vector used in the prediction of the luma prediction block containing the sample p_0 and the list 1 motion vector used in the prediction of the luma prediction block containing the sample q_0 is greater than or equal to 4 in units of quarter luma samples, or the absolute difference between the horizontal or vertical component of the list 1 motion vector used in the prediction of the luma prediction block containing the sample p_0 and list 0 motion vector used in the prediction of the luma prediction block containing the sample q_0 is greater than or equal to 4 in units of quarter luma samples.

– Otherwise, the variable bS[xD_i][yD_j] is set equal to 0.

8.7.2.5 Edge filtering process

8.7.2.5.1 Vertical edge filtering process

Inputs to this process are:

– the picture sample array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$,

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block,

– an array bS specifying the boundary filtering strength.

Outputs of this process are the modified picture sample array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

The filtering process for edges in the luma coding block of the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << (log2CbSize − 3).
2. For xD_k equal to k << 3 with k = 0..nD − 1 and yD_m equal to m << 2 with m = 0..nD * 2 − 1, the following applies:
   – When bS[xD_k][yD_m] is greater than 0, the following ordered steps apply:
      a. The decision process for luma block edges as specified in clause 8.7.2.5.3 is invoked with the luma picture sample array $recPicture_L$, the location of the luma coding block (xCb, yCb), the luma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_VER and the boundary filtering strength bS[xD_k][yD_m] as inputs, and the decisions dE, dEp and dEq, and the variables β and t_C as outputs.
      b. The filtering process for luma block edges as specified in clause 8.7.2.5.4 is invoked with the luma picture sample array $recPicture_L$, the location of the luma coding block (xCb, yCb), the luma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_VER, the decisions dE, dEp and dEq, and the variables β and t_C as inputs, and the modified luma picture sample array $recPicture_L$ as output.

When ChromaArrayType is not equal to 0, the following applies.

The filtering process for edges in the chroma coding blocks of current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << (log2CbSize − 3).
2. The variable edgeSpacing is set equal to 8 / SubWidthC.
3. The variable edgeSections is set equal to nD * (2 / SubHeightC).
4. For xD_k equal to k * edgeSpacing with k = 0..nD − 1 and yD_m equal to m << 2 with m = 0..edgeSections − 1, the following applies:
   – When bS[xD_k * SubWidthC][yD_m * SubHeightC] is equal to 2 and (((xCb / SubWidthC + xD_k) >> 3) << 3) is equal to xCb / SubWidthC + xD_k, the following ordered steps apply:
      a. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array $recPicture_{Cb}$, the location of the chroma coding block (xCb / SubWidthC, yCb / SubHeightC), the chroma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_VER and a variable cQpPicOffset set equal to pps_cb_qp_offset as inputs, and the modified chroma picture sample array $recPicture_{Cb}$ as output.
      b. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array $recPicture_{Cr}$, the location of the chroma coding block (xCb / SubWidthC, yCb / SubHeightC), the chroma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_VER and a variable cQpPicOffset set equal to pps_cr_qp_offset as inputs, and the modified chroma picture sample array $recPicture_{Cr}$ as output.

8.7.2.5.2 Horizontal edge filtering process

Inputs to this process are:

– the picture sample array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$,
– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
– a variable log2CbSize specifying the size of the current luma coding block,
– an array bS specifying the boundary filtering strength.

Outputs of this process are the modified picture sample array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

The filtering process for edges in the luma coding block of the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << (log2CbSize − 3).
2. For yD_m equal to m << 3 with m = 0..nD − 1 and xD_k equal to k << 2 with k = 0..nD * 2 − 1, the following applies:
   – When bS[xD_k][yD_m] is greater than 0, the following ordered steps apply:

a. The decision process for luma block edges as specified in clause 8.7.2.5.3 is invoked with the luma picture sample array $recPicture_L$, the location of the luma coding block (xCb, yCb), the luma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_HOR and the boundary filtering strength bS[xD_k][yD_m] as inputs, and the decisions dE, dEp and dEq, and the variables β and t_C as outputs.

b. The filtering process for luma block edges as specified in clause 8.7.2.5.4 is invoked with the luma picture sample array $recPicture_L$, the location of the luma coding block (xCb, yCb), the luma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_HOR, the decisions dEp, dEp and dEq, and the variables β and t_C as inputs, and the modified luma picture sample array $recPicture_L$ as output.

When ChromaArrayType is not equal to 0, the following applies.

The filtering process for edges in the chroma coding blocks of current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << (log2CbSize − 3).
2. The variable edgeSpacing is set equal to 8 / SubHeightC.
3. The variable edgeSections is set equal to nD * (2 / SubWidthC).
4. For yD_m equal to m * edgeSpacing with m = 0..nD − 1 and xD_k equal to k << 2 with k = 0..edgeSections − 1, the following applies:
   – When bS[xD_k * SubWidthC][yD_m * SubHeightC] is equal to 2 and (((yCb / SubHeightC + yD_m) >> 3) << 3) is equal to yCb / SubHeightC + yD_m, the following ordered steps apply:

     a. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array $recPicture_{Cb}$, the location of the chroma coding block (xCb / SubWidthC, yCb / SubHeightC), the chroma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_HOR and a variable cQpPicOffset set equal to pps_cb_qp_offset as inputs, and the modified chroma picture sample array $recPicture_{Cb}$ as output.

     b. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array $recPicture_{Cr}$, the location of the chroma coding block (xCb / SubWidthC, yCb / SubHeightC), the chroma location of the block (xD_k, yD_m), a variable edgeType set equal to EDGE_HOR and a variable cQpPicOffset set equal to pps_cr_qp_offset as inputs, and the modified chroma picture sample array $recPicture_{Cr}$ as output.

8.7.2.5.3 Decision process for luma block edges

Inputs to this process are:

– a luma picture sample array $recPicture_L$,
– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
– a luma location (xBl, yBl) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,
– a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,
– a variable bS specifying the boundary filtering strength.

Outputs of this process are:

– the variables dE, dEp and dEq containing decisions,
– the variables β and t_C.

If edgeType is equal to EDGE_VER, the sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 0 and 3 are derived as follows:

$$q_{i,k} = recPicture_L[xCb + xBl + i][yCb + yBl + k] \quad (8\text{-}343)$$

$$p_{i,k} = recPicture_L[xCb + xBl - i - 1][yCb + yBl + k] \quad (8\text{-}344)$$

Otherwise (edgeType is equal to EDGE_HOR), the sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 0 and 3 are derived as follows:

$$q_{i,k} = \text{recPicture}_L[\ xCb + xBl + k\ ][\ yCb + yBl + i\ ] \quad (8\text{-}345)$$

$$p_{i,k} = \text{recPicture}_L[\ xCb + xBl + k\ ][\ yCb + yBl - i - 1\ ] \quad (8\text{-}346)$$

The variables Qp_Q and Qp_P are set equal to the Qp_Y values of the coding units which include the coding blocks containing the sample $q_{0,0}$ and $p_{0,0}$, respectively.

The variable qP_L is derived as follows:

$$qP_L = (\ (\ Qp_Q + Qp_P + 1\ ) >> 1\ ) \quad (8\text{-}347)$$

The value of the variable β′ is determined as specified in Table 8-12 based on the luma quantization parameter Q derived as follows:

$$Q = \text{Clip3}(\ 0, 51, qP_L + (\ \text{slice\_beta\_offset\_div2} << 1\ )\ ) \quad (8\text{-}348)$$

where slice_beta_offset_div2 is the value of the syntax element slice_beta_offset_div2 for the slice that contains sample $q_{0,0}$.

The variable β is derived as follows:

$$\beta = \beta' * (\ 1 << (\ \text{BitDepth}_Y - 8\ )\ ) \quad (8\text{-}349)$$

The value of the variable t_C' is determined as specified in Table 8-12 based on the luma quantization parameter Q derived as follows:

$$Q = \text{Clip3}(\ 0, 53, qP_L + 2 * (\ bS - 1\ ) + (\ \text{slice\_tc\_offset\_div2} << 1\ )\ ) \quad (8\text{-}350)$$

where slice_tc_offset_div2 is the value of the syntax element slice_tc_offset_div2 for the slice that contains sample $q_{0,0}$.

The variable t_C is derived as follows:

$$t_C = t_C' * (\ 1 << (\ \text{BitDepth}_Y - 8\ )\ ) \quad (8\text{-}351)$$

Depending on the value of edgeType, the following applies:

– If edgeType is equal to EDGE_VER, the following ordered steps apply:

1. The variables dpq0, dpq3, dp, dq and d are derived as follows:

$$dp0 = \text{Abs}(\ p_{2,0} - 2 * p_{1,0} + p_{0,0}\ ) \quad (8\text{-}352)$$

$$dp3 = \text{Abs}(\ p_{2,3} - 2 * p_{1,3} + p_{0,3}\ ) \quad (8\text{-}353)$$

$$dq0 = \text{Abs}(\ q_{2,0} - 2 * q_{1,0} + q_{0,0}\ ) \quad (8\text{-}354)$$

$$dq3 = \text{Abs}(\ q_{2,3} - 2 * q_{1,3} + q_{0,3}\ ) \quad (8\text{-}355)$$

$$dpq0 = dp0 + dq0 \quad (8\text{-}356)$$

$$dpq3 = dp3 + dq3 \quad (8\text{-}357)$$

$$dp = dp0 + dp3 \quad (8\text{-}358)$$

$$dq = dq0 + dq3 \quad (8\text{-}359)$$

$$d = dpq0 + dpq3 \quad (8\text{-}360)$$

2. The variables dE, dEp and dEq are set equal to 0.

3. When d is less than β, the following ordered steps apply:

a. The variable dpq is set equal to 2 * dpq0.

b. For the sample location (xCb + xBl, yCb + yBl), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,0}$, $p_{3,0}$, $q_{0,0}$, and $q_{3,0}$, the variables dpq, β and t_C as inputs, and the output is assigned to the decision dSam0.

c. The variable dpq is set equal to 2 * dpq3.

d. For the sample location (xCb + xBl, yCb + yBl + 3), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,3}$, $p_{3,3}$, $q_{0,3}$, and $q_{3,3}$, the variables dpq, β and t_C as inputs, and the output is assigned to the decision dSam3.

e. The variable dE is set equal to 1.

f. When dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.

g. When dp is less than (β + (β >> 1)) >> 3, the variable dEp is set equal to 1.

h. When dq is less than (β + (β >> 1)) >> 3, the variable dEq is set equal to 1.

– Otherwise (edgeType is equal to EDGE_HOR), the following ordered steps apply:

1. The variables dpq0, dpq3, dp, dq and d are derived as follows:

$$dp0 = \text{Abs}(p_{2,0} - 2 * p_{1,0} + p_{0,0}) \quad (8\text{-}361)$$

$$dp3 = \text{Abs}(p_{2,3} - 2 * p_{1,3} + p_{0,3}) \quad (8\text{-}362)$$

$$dq0 = \text{Abs}(q_{2,0} - 2 * q_{1,0} + q_{0,0}) \quad (8\text{-}363)$$

$$dq3 = \text{Abs}(q_{2,3} - 2 * q_{1,3} + q_{0,3}) \quad (8\text{-}364)$$

$$dpq0 = dp0 + dq0 \quad (8\text{-}365)$$

$$dpq3 = dp3 + dq3 \quad (8\text{-}366)$$

$$dp = dp0 + dp3 \quad (8\text{-}367)$$

$$dq = dq0 + dq3 \quad (8\text{-}368)$$

$$d = dpq0 + dpq3 \quad (8\text{-}369)$$

2. The variables dE, dEp and dEq are set equal to 0.

3. When d is less than β, the following ordered steps apply:

a. The variable dpq is set equal to 2 * dpq0.

b. For the sample location (xCb + xBl, yCb + yBl), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,0}$, $p_{3,0}$, $q_{0,0}$ and $q_{3,0}$, the variables dpq, β and t_C as inputs, and the output is assigned to the decision dSam0.

c. The variable dpq is set equal to 2 * dpq3.

d. For the sample location (xCb + xBl + 3, yCb + yBl), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,3}$, $p_{3,3}$, $q_{0,3}$ and $q_{3,3}$, the variables dpq, β and t_C as inputs, and the output is assigned to the decision dSam3.

e. The variable dE is set equal to 1.

f. When dSam0 is equal to 1 and dSam3 is equal to 1, the variable dE is set equal to 2.

g. When dp is less than (β + (β >> 1)) >> 3, the variable dEp is set equal to 1.

h. When dq is less than (β + (β >> 1)) >> 3, the variable dEq is set equal to 1.

Table 8-12 – Derivation of threshold variables β′ and t_C′ from input Q

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β′	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C′	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β′	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C′	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53			
β′	38	40	42	44	46	48	50	52	54	56	58	60	62	64	-	-			
t_C′	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24			

8.7.2.5.4 Filtering process for luma block edges

Inputs to this process are:

– a luma picture sample array $recPicture_L$,

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,

– a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,

– the variables dE, dEp and dEq containing decisions,

– the variables β and t_C.

Output of this process is the modified luma picture sample array $recPicture_L$.

Depending on the value of edgeType, the following applies:

– If edgeType is equal to EDGE_VER, the following ordered steps apply:

1. The sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 0..3 are derived as follows:

$$q_{i,k} = recPicture_L[\ xCb + xBl + i\ ][\ yCb + yBl + k\ ] \quad (8\text{-}370)$$

$$p_{i,k} = recPicture_L[\ xCb + xBl - i - 1\ ][\ yCb + yBl + k\ ] \quad (8\text{-}371)$$

2. When dE is not equal to 0, for each sample location (xCb + xBl, yCb + yBl + k), k = 0..3, the following ordered steps apply:

   a. The filtering process for a luma sample as specified in clause 8.7.2.5.7 is invoked with the sample values $p_{i,k}$, $q_{i,k}$ with i = 0..3, the locations (xP_i, yP_i) set equal to (xCb + xBl − i − 1, yCb + yBl + k) and (xQ_i, yQ_i) set equal to (xCb + xBl + i, yCb + yBl + k) with i = 0..2, the decision dE, the variables dEp and dEq and the variable t_C as inputs, and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values p_i' and q_j' as outputs.

   b. When nDp is greater than 0, the filtered sample values p_i' with i = 0..nDp − 1 replace the corresponding samples inside the sample array $recPicture_L$ as follows:

$$recPicture_L[\ xCb + xBl - i - 1\ ][\ yCb + yBl + k\ ] = p_i' \quad (8\text{-}372)$$

   c. When nDq is greater than 0, the filtered sample values q_j' with j = 0..nDq − 1 replace the corresponding samples inside the sample array $recPicture_L$ as follows:

$$recPicture_L[\ xCb + xBl + j\ ][\ yCb + yBl + k\ ] = q_j' \quad (8\text{-}373)$$

– Otherwise (edgeType is equal to EDGE_HOR), the following ordered steps apply:

1. The sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 0..3 are derived as follows:

$$q_{i,k} = \text{recPicture}_L[\ xCb + xBl + k\ ][\ yCb + yBl + i\ ] \quad (8\text{-}374)$$

$$p_{i,k} = \text{recPicture}_L[\ xCb + xBl + k\ ][\ yCb + yBl - i - 1\ ] \quad (8\text{-}375)$$

2. When dE is not equal to 0, for each sample location (xCb + xBl + k, yCb + yBl), k = 0..3, the following ordered steps apply:

   a. The filtering process for a luma sample as specified in clause 8.7.2.5.7 is invoked with the sample values $p_{i,k}$, $q_{i,k}$ with i = 0..3, the locations (xP_i, yP_i) set equal to (xCb + xBl + k, yCb + yBl − i − 1) and (xQ_i, yQ_i) set equal to (xCb + xBl + k, yCb + yBl + i) with i = 0..2, the decision dE, the variables dEp and dEq, and the variable t_C as inputs, and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values p_i' and q_j' as outputs.

   b. When nDp is greater than 0, the filtered sample values p_i' with i = 0..nDp − 1 replace the corresponding samples inside the sample array recPicture_L as follows:

$$\text{recPicture}_L[\ xCb + xBl + k\ ][\ yCb + yBl - i - 1\ ] = p_i' \quad (8\text{-}376)$$

   c. When nDq is greater than 0, the filtered sample values q_j' with j = 0..nDq − 1 replace the corresponding samples inside the sample array recPicture_L as follows:

$$\text{recPicture}_L[\ xCb + xBl + k\ ][\ yCb + yBl + j\ ] = q_j' \quad (8\text{-}377)$$

8.7.2.5.5 Filtering process for chroma block edges

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

– a chroma picture sample array s′,

– a chroma location (xCb, yCb) specifying the top-left sample of the current chroma coding block relative to the top-left chroma sample of the current picture,

– a chroma location (xBl, yBl) specifying the top-left sample of the current chroma block relative to the top-left sample of the current chroma coding block,

– a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,

– a variable cQpPicOffset specifying the picture-level chroma quantization parameter offset.

Output of this process is the modified chroma picture sample array s′.

If edgeType is equal to EDGE_VER, the values p_i and q_i with i = 0..1 and k = 0..3 are derived as follows:

$$q_{i,k} = s'[\ xCb + xBl + i\ ][\ yCb + yBl + k\ ] \quad (8\text{-}378)$$

$$p_{i,k} = s'[\ xCb + xBl - i - 1\ ][\ yCb + yBl + k\ ] \quad (8\text{-}379)$$

Otherwise (edgeType is equal to EDGE_HOR), the sample values p_i and q_i with i = 0..1 and k = 0..3 are derived as follows:

$$q_{i,k} = s'[\ xCb + xBl + k\ ][\ yCb + yBl + i\ ] \quad (8\text{-}380)$$

$$p_{i,k} = s'[\ xCb + xBl + k\ ][\ yCb + yBl - i - 1\ ] \quad (8\text{-}381)$$

The variables Qp_Q and Qp_P are set equal to the Qp_Y values of the coding units which include the coding blocks containing the sample $q_{0,0}$ and $p_{0,0}$, respectively.

The index qPi is derived as follows:

$$qPi = (\ (\ Qp_Q + Qp_P + 1\ ) >> 1\ ) + cQpPicOffset \quad (8\text{-}382)$$

The variable Qp_C is derived as follows:

– If ChromaArrayType is equal to 1, the variable Qp_C is determined based on qPi as specified in Table 8-10.

– Otherwise (ChromaArrayType is greater than 1), the variable Qp_C is set equal to Min(qPi, 51).

NOTE – The variable cQpPicOffset provides an adjustment for the value of pps_cb_qp_offset or pps_cr_qp_offset, according to whether the filtered chroma component is the Cb or Cr component. However, to avoid the need to vary the amount of the adjustment within the picture, the filtering process does not include an adjustment for the value of slice_cb_qp_offset or slice_cr_qp_offset, nor (when chroma_qp_offset_list_enabled_flag is equal to 1) for the value of $CuQpOffset_{Cb}$ or $CuQpOffset_{Cr}$.

The value of the variable t_C' is determined as specified in Table 8-12 based on the chroma quantization parameter Q derived as follows:

$$Q = \text{Clip3}(\ 0, 53, Qp_C + 2 + (\ \text{slice\_tc\_offset\_div2}\ \ll\ 1\ )\ ) \tag{8-383}$$

where slice_tc_offset_div2 is the value of the syntax element slice_tc_offset_div2 for the slice that contains sample $q_{0,0}$.

The variable t_C is derived as follows:

$$t_C = t_C' * (\ 1\ \ll\ (\ \text{BitDepth}_C - 8\ )\ ) \tag{8-384}$$

Depending on the value of edgeType, the following applies:

– If edgeType is equal to EDGE_VER, for each sample location (xCb + xBl, yCb + yBl + k), k = 0..3, the following ordered steps apply:

  1. The filtering process for a chroma sample as specified in clause 8.7.2.5.8 is invoked with the sample values $p_{i,k}$, $q_{i,k}$, with i = 0..1, the locations (xCb + xBl − 1, yCb + yBl + k) and (xCb + xBl, yCb + yBl + k) and the variable t_C as inputs, and the filtered sample values p_0' and q_0' as outputs.

  2. The filtered sample values p_0' and q_0' replace the corresponding samples inside the sample array s′ as follows:

$$s'[\ xCb + xBl\ ][\ yCb + yBl + k\ ] = q_0' \tag{8-385}$$

$$s'[\ xCb + xBl - 1\ ][\ yCb + yBl + k\ ] = p_0' \tag{8-386}$$

– Otherwise (edgeType is equal to EDGE_HOR), for each sample location (xCb + xBl + k, yCb + yBl), k = 0..3, the following ordered steps apply:

  1. The filtering process for a chroma sample as specified in clause 8.7.2.5.8 is invoked with the sample values $p_{i,k}$, $q_{i,k}$, with i = 0..1, the locations (xCb + xBl + k, yCb + yBl − 1) and (xCb + xBl + k, yCb + yBl), and the variable t_C as inputs, and the filtered sample values p_0' and q_0' as outputs.

  2. The filtered sample values p_0' and q_0' replace the corresponding samples inside the sample array s′ as follows:

$$s'[\ xCb + xBl + k\ ][\ yCb + yBl\ ] = q_0' \tag{8-387}$$

$$s'[\ xCb + xBl + k\ ][\ yCb + yBl - 1\ ] = p_0' \tag{8-388}$$

8.7.2.5.6 Decision process for a luma sample

Inputs to this process are:

– the sample values p_0, p_3, q_0 and q_3,

– the variables dpq, β and t_C.

Output of this process is the variable dSam containing a decision.

The variable dSam is specified as follows:

– If dpq is less than (β >> 2), Abs($p_3 - p_0$) + Abs($q_0 - q_3$) is less than (β >> 3) and Abs($p_0 - q_0$) is less than (5 * t_C + 1) >> 1, dSam is set equal to 1.

– Otherwise, dSam is set equal to 0.

8.7.2.5.7 Filtering process for a luma sample

Inputs to this process are:

– the luma sample values p_i and q_i with i = 0..3,

– the luma locations of p_i and q_i, (xP_i, yP_i) and (xQ_i, yQ_i) with i = 0..2,

– a variable dE,

– the variables dEp and dEq containing decisions to filter samples p1 and q1, respectively,
– a variable t_C.

Outputs of this process are:

– the number of filtered samples nDp and nDq,
– the filtered sample values p_i' and q_j' with i = 0..nDp − 1, j = 0..nDq − 1.

Depending on the value of dE, the following applies:

– If the variable dE is equal to 2, nDp and nDq are both set equal to 3 and the following strong filtering applies:

$$p_0' = \text{Clip3}(\ p_0 - 2 * t_C, p_0 + 2 * t_C, (\ p_2 + 2 * p_1 + 2 * p_0 + 2 * q_0 + q_1 + 4\ ) >> 3\ ) \quad (8\text{-}389)$$

$$p_1' = \text{Clip3}(\ p_1 - 2 * t_C, p_1 + 2 * t_C, (\ p_2 + p_1 + p_0 + q_0 + 2\ ) >> 2\ ) \quad (8\text{-}390)$$

$$p_2' = \text{Clip3}(\ p_2 - 2 * t_C, p_2 + 2*t_C, (\ 2 * p_3 + 3 * p_2 + p_1 + p_0 + q_0 + 4\ ) >> 3\ ) \quad (8\text{-}391)$$

$$q_0' = \text{Clip3}(\ q_0 - 2 * t_C, q_0 + 2 * t_C, (\ p_1 + 2 * p_0 + 2 * q_0 + 2 * q_1 + q_2 + 4\ ) >> 3\ ) \quad (8\text{-}392)$$

$$q_1' = \text{Clip3}(\ q_1 - 2 * t_C, q_1 + 2 * t_C, (\ p_0 + q_0 + q_1 + q_2 + 2\ ) >> 2\ ) \quad (8\text{-}393)$$

$$q_2' = \text{Clip3}(\ q_2 - 2 * t_C, q_2 + 2 * t_C, (\ p_0 + q_0 + q_1 + 3 * q_2 + 2 * q_3 + 4\ ) >> 3\ ) \quad (8\text{-}394)$$

– Otherwise, nDp and nDq are set both equal to 0 and the following weak filtering applies:

  – The following applies:

$$\Delta = (\ 9 * (\ q_0 - p_0\ ) - 3 * (\ q_1 - p_1\ ) + 8\ ) >> 4 \quad (8\text{-}395)$$

  – When Abs(Δ) is less than t_C * 10, the following ordered steps apply:

  1. The filtered sample values p_0' and q_0' are specified as follows:

$$\Delta = \text{Clip3}(\ -t_C, t_C, \Delta\ ) \quad (8\text{-}396)$$

$$p_0' = \text{Clip1}_Y(\ p_0 + \Delta\ ) \quad (8\text{-}397)$$

$$q_0' = \text{Clip1}_Y(\ q_0 - \Delta\ ) \quad (8\text{-}398)$$

  2. When dEp is equal to 1, the filtered sample value p_1' is specified as follows:

$$\Delta p = \text{Clip3}(\ -(\ t_C >> 1\ ), t_C >> 1, (\ (\ (\ p_2 + p_0 + 1\ ) >> 1\ ) - p_1 + \Delta\ ) >> 1\ ) \quad (8\text{-}399)$$

$$p_1' = \text{Clip1}_Y(\ p_1 + \Delta p\ ) \quad (8\text{-}400)$$

  3. When dEq is equal to 1, the filtered sample value q_1' is specified as follows:

$$\Delta q = \text{Clip3}(\ -(\ t_C >> 1\ ), t_C >> 1, (\ (\ (\ q_2 + q_0 + 1\ ) >> 1\ ) - q_1 - \Delta\ ) >> 1\ ) \quad (8\text{-}401)$$

$$q_1' = \text{Clip1}_Y(\ q_1 + \Delta q\ ) \quad (8\text{-}402)$$

  4. nDp is set equal to dEp + 1 and nDq is set equal to dEq + 1.

When nDp is greater than 0 and one or more of the following conditions are true, nDp is set equal to 0:

– pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xP_0][yP_0] is equal to 1.
– cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample p_0 is equal to 1.
– palette_mode_flag of the coding unit that includes the coding block containing the sample p0 is equal to 1.

When nDq is greater than 0 and one or more of the following conditions are true, nDq is set equal to 0:

– pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xQ_0][yQ_0] is equal to 1.

– cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample q_0 is equal to 1.

– palette_mode_flag of the coding unit that includes the coding block containing the sample q0 is equal to 1.

8.7.2.5.8 Filtering process for a chroma sample

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

– the chroma sample values p_i and q_i with i = 0..1,

– the chroma locations of p_0 and q_0, (xP_0, yP_0) and (xQ_0, yQ_0),

– a variable t_C.

Outputs of this process are the filtered sample values p_0' and q_0'.

The filtered sample values p_0' and q_0' are derived as follows:

$$\Delta = \mathrm{Clip3}(-t_C, t_C, ((((q_0 - p_0) << 2) + p_1 - q_1 + 4) >> 3)) \quad (8\text{-}403)$$

$$p_0' = \mathrm{Clip1}_C(p_0 + \Delta) \quad (8\text{-}404)$$

$$q_0' = \mathrm{Clip1}_C(q_0 - \Delta) \quad (8\text{-}405)$$

When one or more of the following conditions are true, the filtered sample value, p_0' is substituted by the corresponding input sample value p_0:

– pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xP_0 * SubWidthC][yP_0 * SubHeightC] is equal to 1.

– cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample p_0 is equal to 1.

– palette_mode_flag of the coding unit that includes the coding block containing the sample p0 is equal to 1.

When one or more of the following conditions are true, the filtered sample value, q_0' is substituted by the corresponding input sample value q_0:

– pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xQ_0 * SubWidthC][yQ_0 * SubHeightC] is equal to 1.

– cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample q_0 is equal to 1.

– palette_mode_flag of the coding unit that includes the coding block containing the sample q0 is equal to 1.

8.7.3 Sample adaptive offset process

8.7.3.1 General

Inputs to this process are the reconstructed picture sample array prior to sample adaptive offset $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

Outputs of this process are the modified reconstructed picture sample array after sample adaptive offset $saoPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $saoPicture_{Cb}$ and $saoPicture_{Cr}$.

This process is performed on a CTB basis after the completion of the deblocking filter process for the decoded picture.

The sample values in the modified reconstructed picture sample array $saoPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $saoPicture_{Cb}$ and $saoPicture_{Cr}$ are initially set equal to the sample values in the reconstructed picture sample array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$, respectively.

For every CTU with CTB location (rx, ry), where rx = 0..PicWidthInCtbsY − 1 and ry = 0..PicHeightInCtbsY − 1, the following applies:

– When slice_sao_luma_flag of the current slice is equal to 1, the CTB modification process as specified in clause 8.7.3.2 is invoked with recPicture set equal to $recPicture_L$, cIdx set equal to 0, (rx, ry), and both nCtbSw and nCtbSh set equal to CtbSizeY as inputs, and the modified luma picture sample array $saoPicture_L$ as output.

– When ChromaArrayType is not equal to 0 and slice_sao_chroma_flag of the current slice is equal to 1, the CTB modification process as specified in clause 8.7.3.2 is invoked with recPicture set equal to $recPicture_{Cb}$, cIdx set equal

to 1, (rx, ry), nCtbSw set equal to (1 << CtbLog2SizeY) / SubWidthC and nCtbSh set equal to (1 << CtbLog2SizeY) / SubHeightC as inputs, and the modified chroma picture sample array $saoPicture_{Cb}$ as output.

– When ChromaArrayType is not equal to 0 and slice_sao_chroma_flag of the current slice is equal to 1, the CTB modification process as specified in clause 8.7.3.2 is invoked with recPicture set equal to $recPicture_{Cr}$, cIdx set equal to 2, (rx, ry), nCtbSw set equal to (1 << CtbLog2SizeY) / SubWidthC and nCtbSh set equal to (1 << CtbLog2SizeY) / SubHeightC as inputs, and the modified chroma picture sample array $saoPicture_{Cr}$ as output.

8.7.3.2 CTB modification process

Inputs to this process are:

– the picture sample array recPicture for the colour component cIdx,

– a variable cIdx specifying the colour component index,

– a pair of variables (rx, ry) specifying the CTB location,

– the CTB width nCtbSw and height nCtbSh.

Output of this process is a modified picture sample array saoPicture for the colour component cIdx.

The variable bitDepth is derived as follows:

– If cIdx is equal to 0, bitDepth is set equal to $BitDepth_Y$.

– Otherwise, bitDepth is set equal to $BitDepth_C$.

The location (xCtb, yCtb), specifying the top-left sample of the current CTB for the colour component cIdx relative to the top-left sample of the current picture component cIdx, is derived as follows:

$$(xCtb, yCtb) = (rx * nCtbSw, ry * nCtbSh) \tag{8-406}$$

The sample locations inside the current CTB are derived as follows:

$$(xS_i, yS_j) = (xCtb + i, yCtb + j) \tag{8-407}$$

$$(xY_i, yY_j) = (cIdx == 0) ? (xS_i, yS_j) : (xS_i * SubWidthC, yS_j * SubHeightC) \tag{8-408}$$

For all sample locations (xS_i, yS_j) and (xY_i, yY_j) with i = 0..nCtbSw − 1 and j = 0..nCtbSh − 1, depending on the values of pcm_loop_filter_disabled_flag, pcm_flag[xY_i][yY_j] and cu_transquant_bypass_flag of the coding unit which includes the coding block covering recPicture[xS_i][yS_j], the following applies:

– If one or more of the following conditions are true, saoPicture[xS_i][yS_j] is not modified:

  – pcm_loop_filter_disabled_flag and pcm_flag[xY_i][yY_j] are both equal to 1.

  – cu_transquant_bypass_flag is equal to 1.

  – SaoTypeIdx[cIdx][rx][ry] is equal to 0.

– Otherwise, if SaoTypeIdx[cIdx][rx][ry] is equal to 2, the following ordered steps apply:

  1. The values of hPos[k] and vPos[k] for k = 0..1 are specified in Table 8-13 based on SaoEoClass[cIdx][rx][ry].

  2. The variable edgeIdx is derived as follows:

     – The modified sample locations (xS_{ik}', yS_{jk}') and (xY_{ik}', yY_{jk}') are derived as follows:

$$(xS_{ik}', yS_{jk}') = (xS_i + hPos[k], yS_j + vPos[k]) \tag{8-409}$$

$$(xY_{ik}', yY_{jk}') = (cIdx == 0) ? (xS_{ik}', yS_{jk}') : (xS_{ik}' * SubWidthC, yS_{jk}' * SubHeightC) \tag{8-410}$$

     – If one or more of the following conditions for all sample locations (xS_{ik}', yS_{jk}') and (xY_{ik}', yY_{jk}') with k = 0..1 are true, edgeIdx is set equal to 0:

       – The sample at location (xS_{ik}', yS_{jk}') is outside the picture boundaries.

       – The sample at location (xS_{ik}', yS_{jk}') belongs to a different slice and one of the following two conditions is true:

– $\text{MinTbAddrZs}[xY_{ik}' >> \text{MinTbLog2SizeY}][yY_{jk}' >> \text{MinTbLog2SizeY}]$ is less than $\text{MinTbAddrZs}[xY_i >> \text{MinTbLog2SizeY}][yY_j >> \text{MinTbLog2SizeY}]$ and slice_loop_filter_across_slices_enabled_flag in the slice which the sample $\text{recPicture}[xS_i][yS_j]$ belongs to is equal to 0.

– $\text{MinTbAddrZs}[xY_i >> \text{MinTbLog2SizeY}][yY_j >> \text{MinTbLog2SizeY}]$ is less than $\text{MinTbAddrZs}[xY_{ik}' >> \text{MinTbLog2SizeY}][yY_{jk}' >> \text{MinTbLog2SizeY}]$ and slice_loop_filter_across_slices_enabled_flag in the slice which the sample $\text{recPicture}[xS_{ik}'][yS_{jk}']$ belongs to is equal to 0.

– loop_filter_across_tiles_enabled_flag is equal to 0 and the sample at location (xS_{ik}', yS_{jk}') belongs to a different tile.

– Otherwise, edgeIdx is derived as follows:

– The following applies:

$$\text{edgeIdx} = 2 + \text{Sign}(\text{recPicture}[xS_i][yS_j] - \text{recPicture}[xS_i + \text{hPos}[0]][yS_j + \text{vPos}[0]]) + \text{Sign}(\text{recPicture}[xS_i][yS_j] - \text{recPicture}[xS_i + \text{hPos}[1]][yS_j + \text{vPos}[1]]) \quad (8\text{-}411)$$

– When edgeIdx is equal to 0, 1, or 2, edgeIdx is modified as follows:

$$\text{edgeIdx} = (\text{edgeIdx} == 2) \; ? \; 0 : (\text{edgeIdx} + 1) \quad (8\text{-}412)$$

3. The modified picture sample array $\text{saoPicture}[xS_i][yS_j]$ is derived as follows:

$$\text{saoPicture}[xS_i][yS_j] = \text{Clip3}(0, (1 << \text{bitDepth}) - 1, \text{recPicture}[xS_i][yS_j] + \text{SaoOffsetVal}[\text{cIdx}][\text{rx}][\text{ry}][\text{edgeIdx}]) \quad (8\text{-}413)$$

– Otherwise (SaoTypeIdx[cIdx][rx][ry] is equal to 1), the following ordered steps apply:

1. The variable bandShift is set equal to bitDepth − 5.
2. The variable saoLeftClass is set equal to sao_band_position[cIdx][rx][ry].
3. The list bandTable is defined with 32 elements and all elements are initially set equal to 0. Then, four of its elements (indicating the starting position of bands for explicit offsets) are modified as follows:

```
for( k = 0; k < 4; k++ )
   bandTable[ ( k + saoLeftClass ) & 31 ] = k + 1                    (8-414)
```

4. The variable bandIdx is set equal to $\text{bandTable}[\text{recPicture}[xS_i][yS_j] >> \text{bandShift}]$.
5. The modified picture sample array $\text{saoPicture}[xS_i][yS_j]$ is derived as follows:

$$\text{saoPicture}[xS_i][yS_j] = \text{Clip3}(0, (1 << \text{bitDepth}) - 1, \text{recPicture}[xS_i][yS_j] + \text{SaoOffsetVal}[\text{cIdx}][\text{rx}][\text{ry}][\text{bandIdx}]) \quad (8\text{-}415)$$

Table 8-13 – Specification of hPos and vPos according to the sample adaptive offset class

SaoEoClass[cIdx][rx][ry]	**0**	**1**	**2**	**3**
hPos[0]	−1	0	−1	1
hPos[1]	1	0	1	−1
vPos[0]	0	−1	−1	−1
vPos[1]	0	1	1	1

9 Parsing process

9.1 General

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

This process is invoked when the descriptor of a syntax element in the syntax tables is equal to ue(v), se(v) (see clause 9.2), or ae(v) (see clause 9.3).

9.2 Parsing process for 0-th order Exp-Golomb codes

9.2.1 General

This process is invoked when the descriptor of a syntax element in the syntax tables is equal to ue(v) or se(v).

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

Syntax elements coded as ue(v) or se(v) are Exp-Golomb-coded. The parsing process for these syntax elements begins with reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process is specified as follows:

```
leadingZeroBits = −1
for( b = 0; !b; leadingZeroBits++ )                                   (9-1)
    b = read_bits( 1 )
```

The variable codeNum is then assigned as follows:

$$\text{codeNum} = 2^{\text{leadingZeroBits}} - 1 + \text{read\_bits(leadingZeroBits)} \quad (9\text{-}2)$$

where the value returned from read_bits(leadingZeroBits) is interpreted as a binary representation of an unsigned integer with most significant bit written first.

Table 9-1 illustrates the structure of the Exp-Golomb code by separating the bit string into "prefix" and "suffix" bits. The "prefix" bits are those bits that are parsed as specified above for the computation of leadingZeroBits, and are shown as either 0 or 1 in the bit string column of Table 9-1. The "suffix" bits are those bits that are parsed in the computation of codeNum and are shown as x_i in Table 9-1, with i in the range of 0 to leadingZeroBits − 1, inclusive. Each x_i is equal to either 0 or 1.

Table 9-1 – Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)

Bit string form	Range of codeNum
1	0
0 1 x_0	1..2
0 0 1 x_1 x_0	3..6
0 0 0 1 x_2 x_1 x_0	7..14
0 0 0 0 1 x_3 x_2 x_1 x_0	15..30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31..62
...	...

Table 9-2 illustrates explicitly the assignment of bit strings to codeNum values.

Table 9-2 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)

Bit string	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	...

Depending on the descriptor, the value of a syntax element is derived as follows:

– If the syntax element is coded as ue(v), the value of the syntax element is equal to codeNum.

– Otherwise (the syntax element is coded as se(v)), the value of the syntax element is derived by invoking the mapping process for signed Exp-Golomb codes as specified in clause 9.2.2 with codeNum as input.

9.2.2 Mapping process for signed Exp-Golomb codes

Input to this process is codeNum as specified in clause 9.2.

Output of this process is a value of a syntax element coded as se(v).

The syntax element is assigned to the codeNum by ordering the syntax element by its absolute value in increasing order and representing the positive value for a given absolute value with the lower codeNum. Table 9-3 provides the assignment rule.

Table 9-3 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v)

codeNum	syntax element value
0	0
1	1
2	−1
3	2
4	−2
5	3
6	−3
k	$(-1)^{k+1}$ Ceil(k ÷ 2)

9.3 CABAC parsing process for slice segment data

9.3.1 General

This process is invoked when parsing syntax elements with descriptor ae(v) in clauses 7.3.8.1 through 7.3.8.12.

Inputs to this process are a request for a value of a syntax element and values of prior parsed syntax elements.

Output of this process is the value of the syntax element.

The initialization process as specified in clause 9.3.2 is invoked when starting the parsing of one or more of the following:

1. The slice segment data syntax specified in clause 7.3.8.1,
2. The CTU syntax specified in clause 7.3.8.2 and the CTU is the first CTU in a tile,
3. The CTU syntax specified in clause 7.3.8.2, entropy_coding_sync_enabled_flag is equal to 1 and the associated luma CTB is the first luma CTB in a CTU row of a tile.

The parsing of syntax elements proceeds as follows:

When cabac_bypass_alignment_enabled_flag is equal to 1, the request for a value of a syntax element is for either the syntax elements coeff_abs_level_remaining[ ] or coeff_sign_flag[ ] and escapeDataPresent is equal to 1, the alignment process prior to aligned bypass decoding as specified in clause 9.3.4.3.6 is invoked.

For each requested value of a syntax element a binarization is derived as specified in clause 9.3.3.

The binarization for the syntax element and the sequence of parsed bins determines the decoding process flow as described in clause 9.3.4.

In case the request for a value of a syntax element is processed for the syntax element pcm_flag and the decoded value of pcm_flag is equal to 1, the decoding engine is initialized after the decoding of any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data as specified in clause 9.3.2.6.

The storage process for context variables is applied as follows:

– When ending the parsing of the CTU syntax in clause 7.3.8.2, entropy_coding_sync_enabled_flag is equal to 1 and either CtbAddrInRs % PicWidthInCtbsY is equal to 1 or both CtbAddrInRs is greater than 1 and TileId[CtbAddrInTs] is not equal to TileId[CtbAddrRsToTs[CtbAddrInRs − 2]], the storage process for context variables, Rice parameter initialization states, and palette predictor variables as specified in clause 9.3.2.4 is invoked with TableStateIdxWpp, TableMpsValWpp, TableStatCoeffWpp when persistent_rice_adaptation_enabled_flag is equal to 1, and PredictorPaletteSizeWpp and PredictorPaletteEntriesWpp when palette_mode_enabled_flag is equal to 1 as outputs.
– When ending the parsing of the general slice segment data syntax in clause 7.3.8.1, dependent_slice_segments_enabled_flag is equal to 1 and end_of_slice_segment_flag is equal to 1, the storage process for context variables, Rice parameter initialization states, and palette predictor variables as specified in clause 9.3.2.4 is invoked with TableStateIdxDs, TableMpsValDs, TableStatCoeffDs when persistent_rice_adaptation_enabled_flag is equal to 1, and PredictorPaletteSizeDs and PredictorPaletteEntriesDs when palette_mode_enabled_flag is equal to 1 as outputs.

The whole CABAC parsing process for a syntax element synEl is illustrated in Figure 9-1.

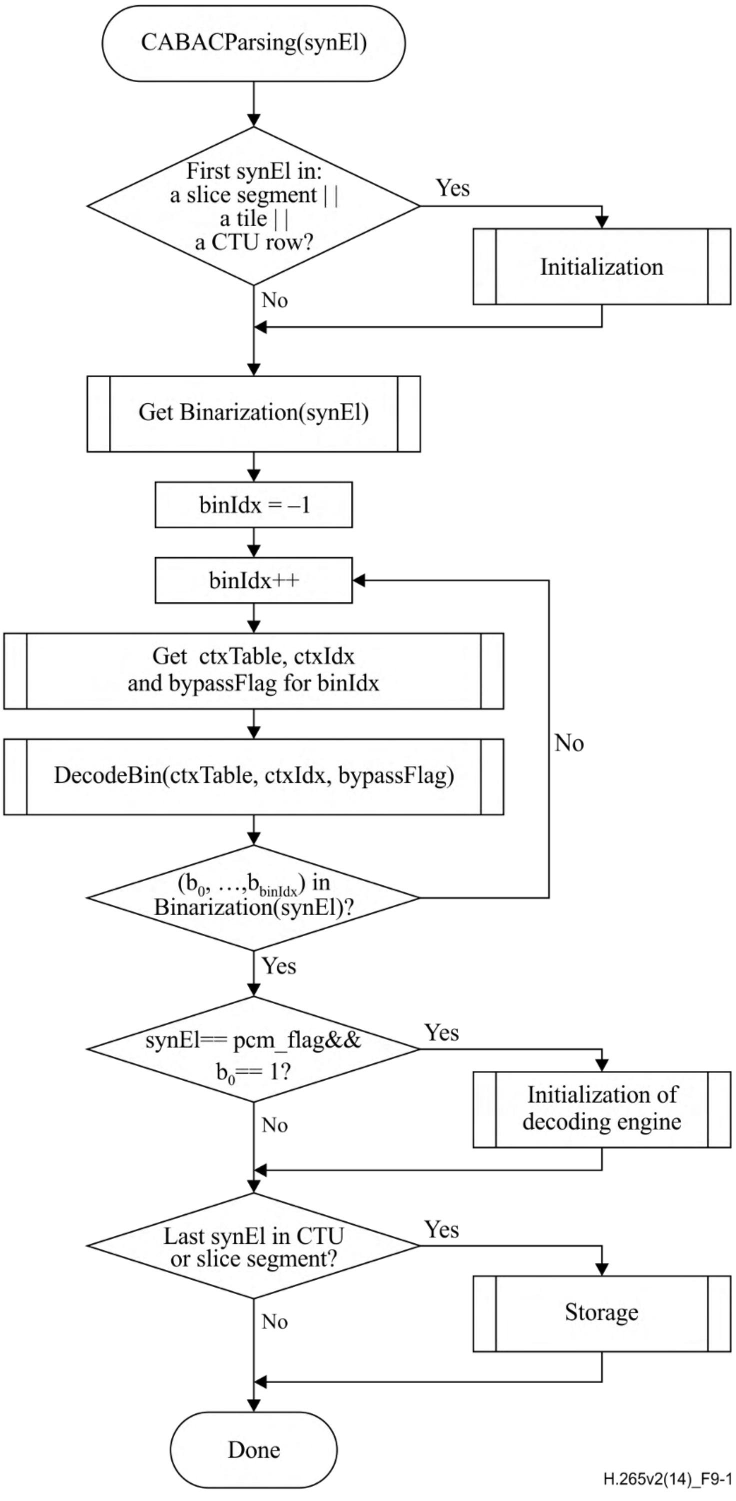


Figure 9-1 – Illustration of CABAC parsing process for a syntax element synEl (informative)

9.3.2 Initialization process

9.3.2.1 General

Outputs of this process are initialized CABAC internal variables, the initialized Rice parameter initialization states StatCoeff, and the initialized palette predictor variables.

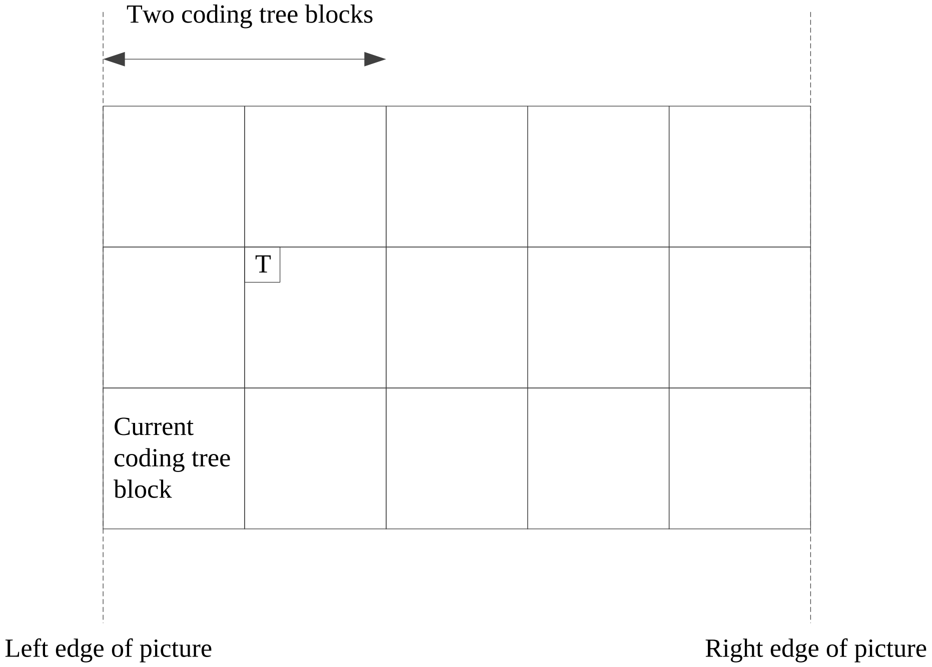


Figure 9-2 – Spatial neighbour T that is used to invoke the CTB availability derivation process relative to the current CTB (informative)

The context variables of the arithmetic decoding engine, Rice parameter initialization states, and palette predictor variables are initialized as follows:

– If the CTU is the first CTU in a tile, the following applies:
  – The initialization process for context variables is invoked as specified in clause 9.3.2.2.
  – The variables StatCoeff[k] are set equal to 0, for k in the range 0 to 3, inclusive.
  – The initialization process for palette predictor variables is invoked as specified in clause 9.3.2.3.
– Otherwise, if entropy_coding_sync_enabled_flag is equal to 1 and either CtbAddrInRs % PicWidthInCtbsY is equal to 0 or TileId[CtbAddrInTs] is not equal to TileId[CtbAddrRsToTs[CtbAddrInRs − 1]], the following applies:
  – The location (xNbT, yNbT) of the top-left luma sample of the spatial neighbouring block T (Figure 9-2) is derived using the location (x0, y0) of the top-left luma sample of the current CTB as follows:

$$(xNbT, yNbT) = (x0 + CtbSizeY, y0 - CtbSizeY) \qquad (9\text{-}3)$$

  – The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbT, yNbT) as inputs, and the output is assigned to availableFlagT.
  – The synchronization process for context variables, Rice parameter initialization states, and palette predictor variables is invoked as follows:
    – If availableFlagT is equal to 1, the synchronization process for context variables, Rice parameter initialization states, and palette predictor variables as specified in clause 9.3.2.5 is invoked with TableStateIdxWpp, TableMpsValWpp, TableStatCoeffWpp, PredictorPaletteSizeWpp, and TablePredictorPaletteEntriesWpp as inputs.
    – Otherwise, the following applies:
      – The initialization process for context variables is invoked as specified in clause 9.3.2.2.

– The variables StatCoeff[k] are set equal to 0, for k in the range 0 to 3, inclusive.

– The initialization process for palette predictor variables is invoked as specified in clause 9.3.2.3.

– Otherwise, if CtbAddrInRs is equal to slice_segment_address and dependent_slice_segment_flag is equal to 1, the synchronization process for context variables and Rice parameter initialization states as specified in clause 9.3.2.5 is invoked with TableStateIdxDs, TableMpsValDs, TableStatCoeffDs, PredictorPaletteSizeDs, and TablePredictorPaletteEntriesDs as inputs.

– Otherwise, the following applies:

  – The initialization process for context variables is invoked as specified in clause 9.3.2.2.

  – The variables StatCoeff[k] are set equal to 0, for k in the range 0 to 3, inclusive.

  – The initialization process for palette predictor variables is invoked as specified in clause 9.3.2.3.

The initialization process for the arithmetic decoding engine is invoked as specified in clause 9.3.2.6.

The whole initialization process for a syntax element synEl is illustrated in the flowchart of Figure 9-3.

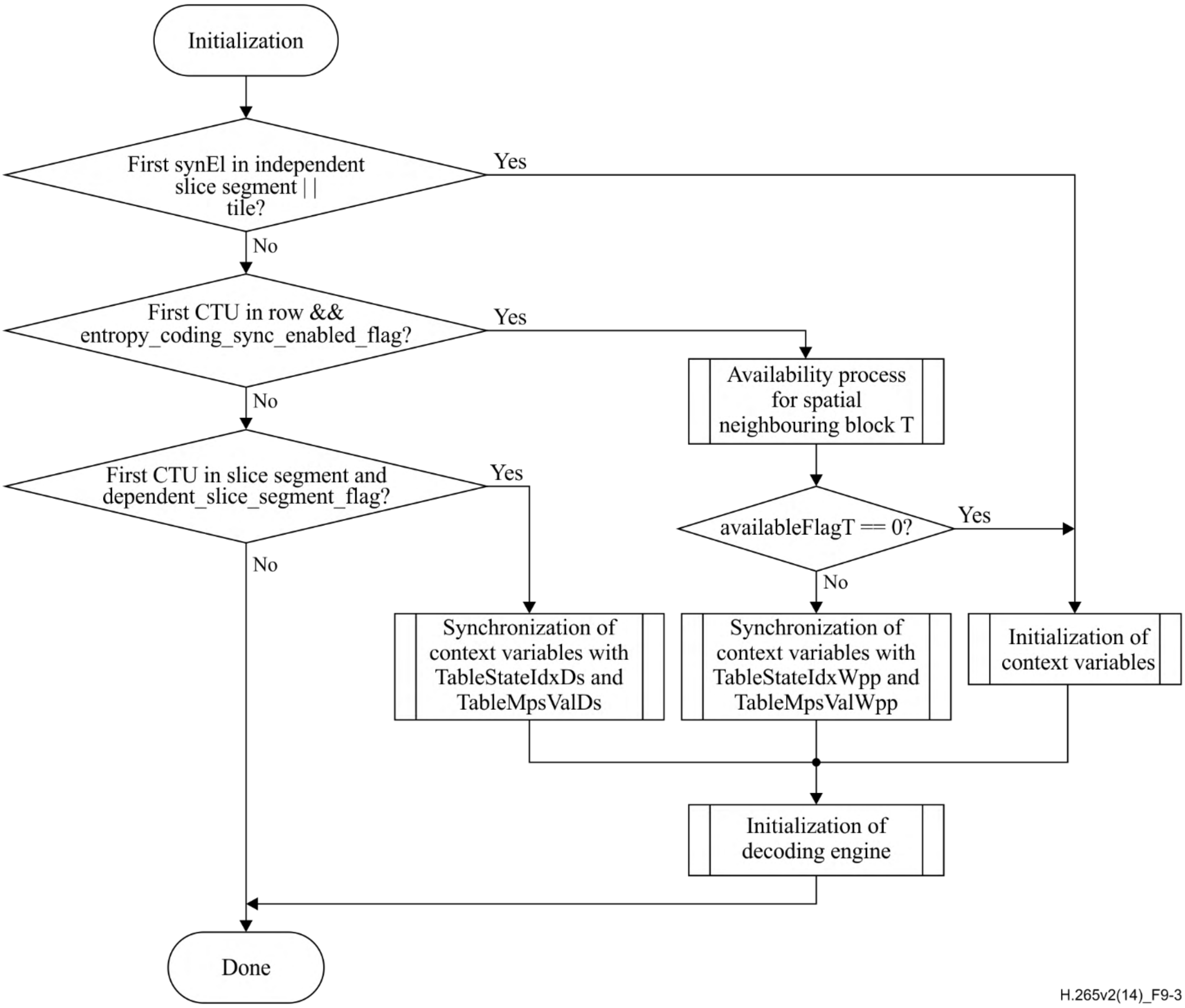


Figure 9-3 – Illustration of CABAC initialization process (informative)

9.3.2.2 Initialization process for context variables

Outputs of this process are the initialized CABAC context variables indexed by ctxTable and ctxIdx.

Table 9-5 to Table 9-37contain the values of the 8 bit variable initValue used in the initialization of context variables that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag.

For each context variable, the two variables pStateIdx and valMps are initialized.

NOTE 1 – The variable pStateIdx corresponds to a probability state index and the variable valMps corresponds to the value of the most probable symbol as further described in clause 9.3.4.3.

From the 8 bit table entry initValue, the two 4 bit variables slopeIdx and offsetIdx are derived as follows:

$$\begin{aligned} &\text{slopeIdx} = \text{initValue} \gg 4 \\ &\text{offsetIdx} = \text{initValue} \,\&\, 15 \end{aligned} \tag{9-4}$$

The variables m and n, used in the initialization of context variables, are derived from slopeIdx and offsetIdx as follows:

$$\begin{aligned} &m = \text{slopeIdx} * 5 - 45 \\ &n = (\text{offsetIdx} \ll 3) - 16 \end{aligned} \tag{9-5}$$

The two values assigned to pStateIdx and valMps for the initialization are derived from SliceQp_Y, which is derived in Equation 7-54. Given the variables m and n, the initialization is specified as follows:

$$\begin{aligned} &\text{preCtxState} = \text{Clip3}(1, 126, ((m * \text{Clip3}(0, 51, \text{SliceQp}_Y)) \gg 4) + n) \\ &\text{valMps} = (\text{preCtxState} <= 63) ? 0 : 1 \\ &\text{pStateIdx} = \text{valMps} ? (\text{preCtxState} - 64) : (63 - \text{preCtxState}) \end{aligned} \tag{9-6}$$

In Table 9-4, the ctxIdx for which initialization is needed for each of the three initialization types, specified by the variable initType, are listed. Also listed is the table number that includes the values of initValue needed for the initialization. For P and B slice types, the derivation of initType depends on the value of the cabac_init_flag syntax element. The variable initType is derived as follows:

```
if( slice_type = = I )
    initType = 0
else if( slice_type = = P )
    initType = cabac_init_flag ? 2 : 1                                  (9-7)
else
    initType = cabac_init_flag ? 1 : 2
```

Table 9-4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

Syntax structure	Syntax element	ctxTable	initType		
			0	**1**	**2**
sao( )	sao_merge_left_flag sao_merge_up_flag	Table 9-5	0	1	2
	sao_type_idx_luma sao_type_idx_chroma	Table 9-6	0	1	2
coding_quadtree( )	split_cu_flag[ ][ ]	Table 9-7	0..2	3..5	6..8
coding_unit( )	cu_transquant_bypass_flag	Table 9-8	0	1	2
	cu_skip_flag	Table 9-9		0..2	3..5
	palette_mode_flag	Table 9-38	0	1	2
	pred_mode_flag	Table 9-10		0	1
	part_mode	Table 9-11	0	1..4	5..8
	prev_intra_luma_pred_flag[ ][ ]	Table 9-12	0	1	2
	intra_chroma_pred_mode[ ][ ]	Table 9-13	0	1	2
	rqt_root_cbf	Table 9-14		0	1
prediction_unit( )	merge_flag[ ][ ]	Table 9-15		0	1

Table 9-4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

Syntax structure	Syntax element	ctxTable	initType 0	initType 1	initType 2
	merge_idx[ ][ ]	Table 9-16		0	1
	inter_pred_idc[ ][ ]	Table 9-17		0..4	5..9
	ref_idx_l0[ ][ ], ref_idx_l1[ ][ ]	Table 9-18		0..1	2..3
	mvp_l0_flag[ ][ ], mvp_l1_flag[ ][ ]	Table 9-19		0	1
transform_tree( )	split_transform_flag[ ][ ][ ]	Table 9-20	0..2	3..5	6..8
	cbf_luma[ ][ ][ ]	Table 9-21	0..1	2..3	4..5
	cbf_cb[ ][ ][ ], cbf_cr[ ][ ][ ]	Table 9-22	0..3 12	4..7 13	8..11 14
mvd_coding( )	abs_mvd_greater0_flag[ ]	Table 9-23		0	2
	abs_mvd_greater1_flag[ ]	Table 9-23		1	3
transform_unit( )	tu_residual_act_flag	Table 9-39	0	1	2
cross_comp_pred( )	log2_res_scale_abs_plus1[ ]	Table 9-36	0..7	8..15	16..23
	res_scale_sign_flag[ ]	Table 9-37	0..1	2..3	4..5
residual_coding( )	transform_skip_flag[ ][ ][0]	Table 9-25	0	1	2
	transform_skip_flag[ ][ ][1] transform_skip_flag[ ][ ][2]	Table 9-25	3	4	5
	explicit_rdpcm_flag[ ][ ][0]	Table 9-32		0	1
	explicit_rdpcm_flag[ ][ ][1] explicit_rdpcm_flag[ ][ ][2]	Table 9-32		2	3
	explicit_rdpcm_dir_flag[ ][ ][0]	Table 9-33		0	1
	explicit_rdpcm_dir_flag[ ][ ][1] explicit_rdpcm_dir_flag[ ][ ][2]	Table 9-33		2	3
	last_sig_coeff_x_prefix	Table 9-26	0..17	18..35	36..53
	last_sig_coeff_y_prefix	Table 9-27	0..17	18..35	36..53
	coded_sub_block_flag[ ][ ]	Table 9-28	0..3	4..7	8..11
	sig_coeff_flag[ ][ ]	Table 9-29	0..41 126..127	42..83 128..129	84..125 130..131
	coeff_abs_level_greater1_flag[ ]	Table 9-30	0..23	24..47	48..71
	coeff_abs_level_greater2_flag[ ]	Table 9-31	0..5	6..11	12..17
palette_coding( )	palette_run_prefix	Table 9-40	0..7	8..15	16..23
	copy_above_palette_indices_flag	Table 9-41	0	1	2
	copy_above_indices_for_final_run_flag	Table 9-41	0	1	2
	palette_transpose_flag	Table 9-42	0	1	2
delta_qp( )	cu_qp_delta_abs	Table 9-24	0..1	2..3	4..5
chroma_qp_offset( )	cu_chroma_qp_offset_flag	Table 9-34	0	1	2
	cu_chroma_qp_offset_idx	Table 9-35	0	1	2

NOTE 2 – ctxTable equal to 0 and ctxIdx equal to 0 are associated with end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag. The decoding process specified in clause 9.3.4.3.5 applies to ctxTable equal to 0 and ctxIdx equal to 0. This decoding process, however, may also be implemented by using the decoding process specified in clause 9.3.4.3.2. In this case, the initial values associated with ctxTable equal to 0 and ctxIdx equal to 0 are specified to be pStateIdx = 63 and valMps = 0, where pStateIdx = 63 represents a non-adapting probability state.

Table 9-5 – Values of initValue for ctxIdx of sao_merge_left_flag and sao_merge_up_flag

Initialization variable	ctxIdx of sao_merge_left_flag and sao_merge_up_flag		
	0	1	2
initValue	153	153	153

Table 9-6 – Values of initValue for ctxIdx of sao_type_idx_luma and sao_type_idx_chroma

Initialization variable	ctxIdx of sao_type_idx_luma and sao_type_idx_chroma		
	0	1	2
initValue	200	185	160

Table 9-7 – Values of initValue for ctxIdx of split_cu_flag

Initialization variable	ctxIdx of split_cu_flag								
	0	1	2	3	4	5	6	7	8
initValue	139	141	157	107	139	126	107	139	126

Table 9-8 – Values of initValue for ctxIdx of cu_transquant_bypass_flag

Initialization variable	ctxIdx of cu_transquant_bypass_flag		
	0	1	2
initValue	154	154	154

Table 9-9 – Values of initValue for ctxIdx of cu_skip_flag

Initialization variable	ctxIdx of cu_skip_flag					
	0	1	2	3	4	5
initValue	197	185	201	197	185	201

Table 9-10 – Values of initValue for ctxIdx of pred_mode_flag

Initialization variable	ctxIdx of pred_mode_flag	
	0	1
initValue	149	134

Table 9-11 – Values of initValue for ctxIdx of part_mode

Initialization variable	ctxIdx of part_mode								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
initValue	184	154	139	154	154	154	139	154	154

Table 9-12 – Values of initValue for ctxIdx of prev_intra_luma_pred_flag

Initialization variable	ctxIdx of prev_intra_luma_pred_flag		
	0	**1**	**2**
initValue	184	154	183

Table 9-13 – Values of initValue for ctxIdx of intra_chroma_pred_mode

Initialization variable	ctxIdx of intra_chroma_pred_mode		
	0	**1**	**2**
initValue	63	152	152

Table 9-14 – Values of initValue for ctxIdx of rqt_root_cbf

Initialization variable	ctxIdx of rqt_root_cbf	
	0	**1**
initValue	79	79

Table 9-15 – Values of initValue for ctxIdx of merge_flag

Initialization variable	ctxIdx of merge_flag	
	0	**1**
initValue	110	154

Table 9-16 – Values of initValue for ctxIdx of merge_idx

Initialization variable	ctxIdx of merge_idx	
	0	**1**
initValue	122	137

Table 9-17 – Values of initValue for ctxIdx of inter_pred_idc

Initialization variable	ctxIdx of inter_pred_idc									
	0	1	2	3	4	5	6	7	8	9
initValue	95	79	63	31	31	95	79	63	31	31

Table 9-18 – Values of initValue for ctxIdx of ref_idx_l0 and ref_idx_l1

Initialization variable	ctxIdx of ref_idx_l0 and ref_idx_l1			
	0	1	2	3
initValue	153	153	153	153

Table 9-19 – Values of initValue for ctxIdx of mvp_l0_flag and mvp_l1_flag

Initialization variable	ctxIdx of mvp_l0_flag and mvp_l1_flag	
	0	1
initValue	168	168

Table 9-20 – Values of initValue for ctxIdx of split_transform_flag

Initialization variable	ctxIdx of split_transform_flag								
	0	1	2	3	4	5	6	7	8
initValue	153	138	138	124	138	94	224	167	122

Table 9-21 – Values of initValue for ctxIdx of cbf_luma

Initialization variable	ctxIdx of cbf_luma					
	0	1	2	3	4	5
initValue	111	141	153	111	153	111

Table 9-22 – Values of initValue for ctxIdx of cbf_cb and cbf_cr

Initialization variable	ctxIdx of cbf_cb and cbf_cr														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
initValue	94	138	182	154	149	107	167	154	149	92	167	154	154	154	154

Table 9-23 – Values of initValue for ctxIdx of abs_mvd_greater0_flag and abs_mvd_greater1_flag

Initialization variable	ctxIdx of abs_mvd_greater0_flag and abs_mvd_greater1_flag			
	0	**1**	**2**	**3**
initValue	140	198	169	198

Table 9-24 – Values of initValue for ctxIdx of cu_qp_delta_abs

Initialization variable	ctxIdx of cu_qp_delta_abs					
	0	**1**	**2**	**3**	**4**	**5**
initValue	154	154	154	154	154	154

Table 9-25 – Values of initValue for ctxIdx of transform_skip_flag

Initialization variable	ctxIdx of transform_skip_flag					
	0	**1**	**2**	**3**	**4**	**5**
initValue	139	139	139	139	139	139

Table 9-26 – Values of initValue for ctxIdx of last_sig_coeff_x_prefix

Initialization variable	ctxIdx of last_sig_coeff_x_prefix																	
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
initValue	110	110	124	125	140	153	125	127	140	109	111	143	127	111	79	108	123	63
	18	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**
initValue	125	110	94	110	95	79	125	111	110	78	110	111	111	95	94	108	123	108
	36	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**
initValue	125	110	124	110	95	94	125	111	111	79	125	126	111	111	79	108	123	93

Table 9-27 – Values of initValue for ctxIdx of last_sig_coeff_y_prefix

Initialization variable	ctxIdx of last_sig_coeff_y_prefix																	
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
initValue	110	110	124	125	140	153	125	127	140	109	111	143	127	111	79	108	123	63
	18	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**
initValue	125	110	94	110	95	79	125	111	110	78	110	111	111	95	94	108	123	108
	36	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**
initValue	125	110	124	110	95	94	125	111	111	79	125	126	111	111	79	108	123	93

Table 9-28 – Values of initValue for ctxIdx of coded_sub_block_flag

Initialization variable	ctxIdx of coded_sub_block_flag											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
initValue	91	171	134	141	121	140	61	154	121	140	61	154

Table 9-29 – Values of initValue for ctxIdx of sig_coeff_flag

Initialization variable	ctxIdx of sig_coeff_flag															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
initValue	111	111	125	110	110	94	124	108	124	107	125	141	179	153	125	107
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
initValue	125	141	179	153	125	107	125	141	179	153	125	140	139	182	182	152
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
initValue	136	152	136	153	136	139	111	136	139	111	155	154	139	153	139	123
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
initValue	123	63	153	166	183	140	136	153	154	166	183	140	136	153	154	166
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
initValue	183	140	136	153	154	170	153	123	123	107	121	107	121	167	151	183
	80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
initValue	140	151	183	140	170	154	139	153	139	123	123	63	124	166	183	140
	96	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
initValue	136	153	154	166	183	140	136	153	154	166	183	140	136	153	154	170
	112	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**
initValue	153	138	138	122	121	122	121	167	151	183	140	151	183	140	141	111
	128	**129**	**130**	**131**												
initValue	140	140	140	140												

Table 9-30 – Values of initValue for ctxIdx of coeff_abs_level_greater1_flag

Initialization variable	ctxIdx of coeff_abs_level_greater1_flag															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
initValue	140	92	137	138	140	152	138	139	153	74	149	92	139	107	122	152
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
initValue	140	179	166	182	140	227	122	197	154	196	196	167	154	152	167	182
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
initValue	182	134	149	136	153	121	136	137	169	194	166	167	154	167	137	182
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
initValue	154	196	167	167	154	152	167	182	182	134	149	136	153	121	136	122
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**								
initValue	169	208	166	167	154	152	167	182								

Table 9-31 – Values of initValue for ctxIdx of coeff_abs_level_greater2_flag

Initialization variable	ctxIdx of coeff_abs_level_greater2_flag								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
initValue	138	153	136	167	152	152	107	167	91
	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
initValue	122	107	167	107	167	91	107	107	167

Table 9-32 – Values of initValue for ctxIdx of explicit_rdpcm_flag

Initialization variable	ctxIdx of explicit_rdpcm_flag			
	0	**1**	**2**	**3**
initValue	139	139	139	139

Table 9-33 – Values of initValue for ctxIdx of explicit_rdpcm_dir_flag

Initialization variable	ctxIdx of explicit_rdpcm_dir_flag			
	0	**1**	**2**	**3**
initValue	139	139	139	139

Table 9-34 – Values of initValue for ctxIdx of cu_chroma_qp_offset_flag

Initialization variable	ctxIdx of cu_chroma_qp_offset_flag		
	0	**1**	**2**
initValue	154	154	154

Table 9-35 – Values of initValue for ctxIdx of cu_chroma_qp_offset_idx

Initialization variable	ctxIdx of cu_chroma_qp_offset_idx		
	0	**1**	**2**
initValue	154	154	154

Table 9-36 – Values of initValue for ctxIdx of log2_res_scale_abs_plus1

Initialization variable	ctxIdx of log2_res_scale_abs_plus1								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
initValue	154	154	154	154	154	154	154	154	154
	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
initValue	154	154	154	154	154	154	154	154	154
	18	**19**	**20**	**21**	**22**	**23**			
initValue	154	154	154	154	154	154			

Table 9-37 – Values of initValue for ctxIdx of res_scale_sign_flag

Initialization variable	ctxIdx of res_scale_sign_flag					
	0	**1**	**2**	**3**	**4**	**5**
initValue	154	154	154	154	154	154

Table 9-38 – Values of initValue for ctxIdx of palette_mode_flag

Initialization variable	ctxIdx of palette_mode_flag		
	0	**1**	**2**
initValue	154	154	154

Table 9-39 – Values of initValue for ctxIdx of tu_residual_act_flag

Initialization variable	ctxIdx of tu_residual_act_flag		
	0	**1**	**2**
initValue	154	154	154

Table 9-40 – Values of initValue for ctxIdx of palette_run_prefix

Initialization variable	ctxIdx of palette_run_prefix											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
initValue	154	154	154	154	154	154	154	154	154	154	154	154
	12	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
initValue	154	154	154	154	154	154	154	154	154	154	154	154

Table 9-41 – Values of initValue for ctxIdx of copy_above_palette_indices_flag and copy_above_indices_for_final_run_flag

Initialization variable	ctxIdx of copy_above_palette_indices_flag and copy_above_indices_for_final_run_flag		
	0	**1**	**2**
initValue	154	154	154

Table 9-42 – Values of initValue for ctxIdx of palette_transpose_flag

Initialization variable	ctxIdx of palette_transpose_flag		
	0	**1**	**2**
initValue	154	154	154

9.3.2.3 Initialization process for palette predictor entries

Outputs of this process are the initialized palette predictor variables PredictorPaletteSize and PredictorPaletteEntries.

The variable numComps is derived as follows:

$$\text{numComps} = (\text{ChromaArrayType} == 0)\ ?\ 1 : 3 \quad (9\text{-}8)$$

– If pps_palette_predictor_initializers_present_flag is equal to 1, the following applies:
  – PredictorPaletteSize is set equal to pps_num_palette_predictor_initializers.
  – The array PredictorPaletteEntries is derived as follows:

```
for( comp = 0; comp < numComps; comp++ )
    for( i = 0; i < PredictorPaletteSize; i++ )                                  (9-9)
        PredictorPaletteEntries[ comp ][ i ] = pps_palette_predictor_initializer[ comp ][ i ]
```

– Otherwise (pps_palette_predictor_initializers_present_flag is equal to 0), if sps_palette_predictor_initializers_present_flag is equal to 1, the following applies:
  – PredictorPaletteSize is set equal to sps_num_palette_predictor_initializers_minus1 plus 1.
  – The array PredictorPaletteEntries is derived as follows:

```
for( comp = 0; comp < numComps; comp++ )
    for( i = 0; i < PredictorPaletteSize; i++ )                                  (9-10)
        PredictorPaletteEntries[ comp ][ i ] = sps_palette_predictor_initializer[ comp ][ i ]
```

Otherwise (pps_palette_predictor_initializers_present_flag is equal to 0 and sps_palette_predictor_initializers_present_flag is equal to 0), PredictorPaletteSize is set equal to 0.

9.3.2.4 Storage process for context variables, Rice parameter initialization states, and palette predictor variables

Inputs to this process are:

– The CABAC context variables indexed by ctxTable and ctxIdx.
– The Rice parameter initialization states indexed by k.
– The palette predictor variables, PredictorPaletteSize and PredictorPaletteEntries.

Outputs of this process are:

– The variables tableStateSync and tableMPSSync containing the values of the variables pStateIdx and valMps used in the initialization process of context variables and Rice parameter initialization states that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag.

– The variables tableStatCoeffSync containing the values of the variables StatCoeff[k] used in the initialization process of context variables and Rice parameter initialization states.

– The variables PredictorPaletteSizeSync and tablePredictorPaletteEntriesSync containing the values used in the initialization process of palette predictor variables.

For each context variable, the corresponding entries pStateIdx and valMps of tables tableStateSync and tableMPSSync are initialized to the corresponding pStateIdx and valMps.

For each Rice parameter initialization state k, each entry of the table tableStatCoeffSync is initialized to the corresponding value of StatCoeff[k].

For palette predictor variables, PredictorPaletteSizeSync is initialized to PredictorPaletteSize. For tablePredictorPaletteEntriesSync, each entry is initialized to the corresponding value of PredictorPaletteEntries.

The storage process for context variables is illustrated in the flowchart of Figure 9-4.

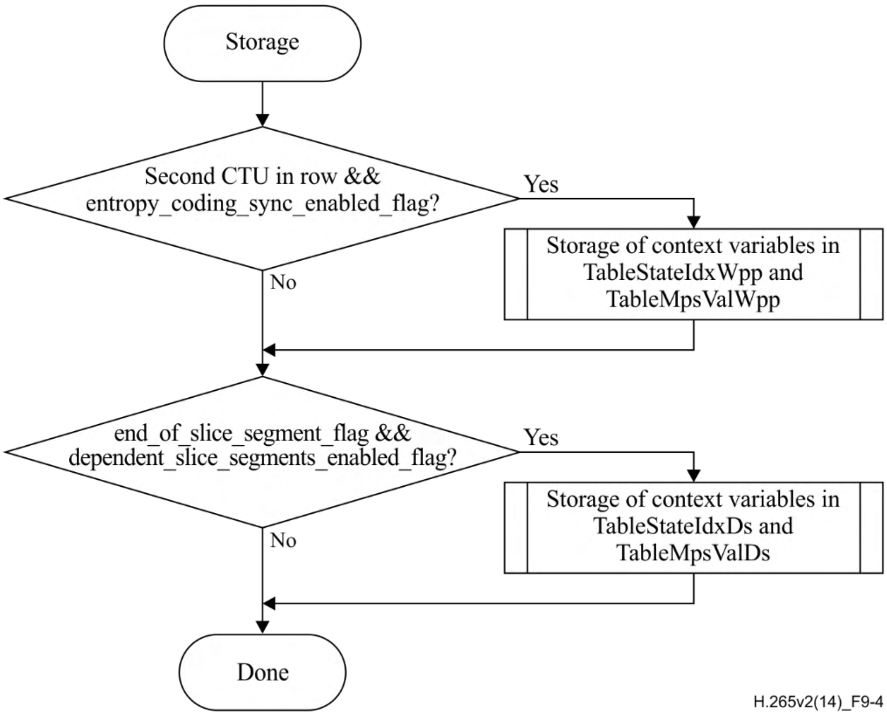


Figure 9-4 – Illustration of CABAC storage process (informative)

9.3.2.5 Synchronization process for context variables, Rice parameter initialization states, and palette predictor variables

Inputs to this process are:

– The variables tableStateSync and tableMPSSync containing the values of the variables pStateIdx and valMps used in the storage process of context variables that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag.

– The variable tableStatCoeffSync containing the values of the variables StatCoeff[k] used in the storage process of context variables and Rice parameter initialization states.

– The variables PredictorPaletteSizeSync and tablePredictorPaletteEntriesSync containing the values used in the storage process of palette predictor variables.

Outputs of this process are:

– The initialized CABAC context variables indexed by ctxTable and ctxIdx.

– The initialized Rice parameter initialization states StatCoeff indexed by k.

– The palette predictor variables, PredictorPaletteSize and PredictorPaletteEntries.

For each context variable, the corresponding context variables pStateIdx and valMps are initialized to the corresponding entries pStateIdx and valMps of tables tableStateSync and tableMPSSync.

For each Rice parameter initialization state, each variable StatCoeff[k] is initialized to the corresponding entry of table tableStatCoeffSync.

For palette predictor variables, PredictorPaletteSize is initialized to PredictorPaletteSizeSync. For PredictorPaletteEntries, each entry is initialized to the corresponding value of tablePredictorPaletteEntriesSync.

9.3.2.6 Initialization process for the arithmetic decoding engine

Outputs of this process are the initialized decoding engine registers ivlCurrRange and ivlOffset both in 16 bit register precision.

The status of the arithmetic decoding engine is represented by the variables ivlCurrRange and ivlOffset. In the initialization procedure of the arithmetic decoding process, ivlCurrRange is set equal to 510 and ivlOffset is set equal to the value returned from read_bits(9) interpreted as a 9 bit binary representation of an unsigned integer with the most significant bit written first.

The bitstream shall not contain data that result in a value of ivlOffset being equal to 510 or 511.

NOTE – The description of the arithmetic decoding engine in this Specification utilizes 16 bit register precision. However, a minimum register precision of 9 bits is required for storing the values of the variables ivlCurrRange and ivlOffset after invocation of the arithmetic decoding process (DecodeBin) as specified in clause 9.3.4.3. The arithmetic decoding process for a binary decision (DecodeDecision) as specified in clause 9.3.4.3.2 and the decoding process for a binary decision before termination (DecodeTerminate) as specified in clause 9.3.4.3.5 require a minimum register precision of 9 bits for the variables ivlCurrRange and ivlOffset. The bypass decoding process for binary decisions (DecodeBypass) as specified in clause 9.3.4.3.4 requires a minimum register precision of 10 bits for the variable ivlOffset and a minimum register precision of 9 bits for the variable ivlCurrRange.

9.3.3 Binarization process

9.3.3.1 General

Input to this process is a request for a syntax element.

Output of this process is the binarization of the syntax element.

Table 9-43 specifies the type of binarization process associated with each syntax element and corresponding inputs.

The specification of the truncated Rice (TR) binarization process, the k-th order Exp-Golomb (EGk) binarization process, limited k-th order Exp-Golomb (EGk) binarization process, the fixed-length (FL) binarization process, and the truncated binary binarization process are given in clauses 9.3.3.2 through 9.3.3.6, respectively. Other binarizations are specified in clauses 9.3.3.7 through 9.3.3.14.

Table 9-43 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
slice_segment_data( )	end_of_slice_segment_flag	FL	cMax = 1
	end_of_subset_one_bit	FL	cMax = 1
sao( )	sao_merge_left_flag	FL	cMax = 1
	sao_merge_up_flag	FL	cMax = 1
	sao_type_idx_luma	TR	cMax = 2, cRiceParam = 0
	sao_type_idx_chroma	TR	cMax = 2, cRiceParam = 0
	sao_offset_abs[ ][ ][ ][ ]	TR	cMax = (1 << (Min(bitDepth, 10) − 5)) − 1, cRiceParam = 0
	sao_offset_sign[ ][ ][ ][ ]	FL	cMax = 1
	sao_band_position[ ][ ][ ]	FL	cMax = 31
	sao_eo_class_luma	FL	cMax = 3
	sao_eo_class_chroma	FL	cMax = 3

Table 9-43 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	**Input parameters**
coding_quadtree( )	split_cu_flag[ ][ ]	FL	cMax = 1
coding_unit( )	cu_transquant_bypass_flag	FL	cMax = 1
	cu_skip_flag	FL	cMax = 1
	palette_mode_flag	FL	cMax = 1
	pred_mode_flag	FL	cMax = 1
	part_mode	9.3.3.7	(xCb, yCb) = (x0, y0), log2CbSize
	pcm_flag[ ][ ]	FL	cMax = 1
	prev_intra_luma_pred_flag[ ][ ]	FL	cMax = 1
	mpm_idx[ ][ ]	TR	cMax = 2, cRiceParam = 0
	rem_intra_luma_pred_mode[ ][ ]	FL	cMax = 31
	intra_chroma_pred_mode[ ][ ]	9.3.3.8	-
	rqt_root_cbf	FL	cMax = 1
prediction_unit( )	merge_flag[ ][ ]	FL	cMax = 1
	merge_idx[ ][ ]	TR	cMax = MaxNumMergeCand − 1, cRiceParam = 0
	inter_pred_idc[x0][y0]	9.3.3.9	nPbW, nPbH
	ref_idx_l0[ ][ ]	TR	cMax = num_ref_idx_l0_active_minus1, cRiceParam = 0
	mvp_l0_flag[ ][ ]	FL	cMax = 1
	ref_idx_l1[ ][ ]	TR	cMax = num_ref_idx_l1_active_minus1, cRiceParam = 0
	mvp_l1_flag[ ][ ]	FL	cMax = 1
transform_tree( )	split_transform_flag[ ][ ][ ]	FL	cMax = 1
	cbf_luma[ ][ ][ ]	FL	cMax = 1
	cbf_cb[ ][ ][ ]	FL	cMax = 1
	cbf_cr[ ][ ][ ]	FL	cMax = 1
mvd_coding( )	abs_mvd_greater0_flag[ ]	FL	cMax = 1
	abs_mvd_greater1_flag[ ]	FL	cMax = 1
	abs_mvd_minus2[ ]	EG1	-
	mvd_sign_flag[ ]	FL	cMax = 1
transform_unit( )	tu_residual_act_flag	FL	cMax = 1

Table 9-43 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	**Input parameters**
cross_comp_pred( )	log2_res_scale_abs_plus1	TR	cMax = 4, cRiceParam = 0
	res_scale_sign_flag	FL	cMax = 1
residual_coding( )	transform_skip_flag[ ][ ][ ]	FL	cMax = 1
	explicit_rdpcm_flag[ ][ ][ ]	FL	cMax = 1
	explicit_rdpcm_dir_flag[ ][ ][ ]	FL	cMax = 1
	last_sig_coeff_x_prefix	TR	cMax = (log2TrafoSize << 1) − 1, cRiceParam = 0
	last_sig_coeff_y_prefix	TR	cMax = (log2TrafoSize << 1) − 1, cRiceParam = 0
	last_sig_coeff_x_suffix	FL	cMax = (1 << ((last_sig_coeff_x_prefix >> 1) − 1) − 1)
	last_sig_coeff_y_suffix	FL	cMax = (1 << ((last_sig_coeff_y_prefix >> 1) − 1) − 1)
	coded_sub_block_flag[ ][ ]	FL	cMax = 1
	sig_coeff_flag[ ][ ]	FL	cMax = 1
	coeff_abs_level_greater1_flag[ ]	FL	cMax = 1
	coeff_abs_level_greater2_flag[ ]	FL	cMax = 1
	coeff_abs_level_remaining[ ]	9.3.3.11	current sub-block scan index i, baseLevel
	coeff_sign_flag[ ]	FL	cMax = 1
palette_coding( )	palette_predictor_run	EG0	-
	num_signalled_palette_entries	EG0	-
	new_palette_entries	FL	cMax = cIdx = = 0 ? ((1 << BitDepth_Y) − 1) : ((1 << BitDepth_C) − 1)
	palette_escape_val_present_flag	FL	cMax = 1
	num_palette_indices_minus1	9.3.3.14	MaxPaletteIndex
	palette_idx_idc	9.3.3.13	MaxPaletteIndex
	copy_above_indices_for_final_run_flag	FL	cMax = 1
	palette_transpose_flag	FL	cMax = 1
	copy_above_palette_indices_flag	FL	cMax = 1
	palette_run_prefix	TR	cMax = Floor(Log2(PaletteMaxRunMinus1)) + 1, cRiceParam = 0
	palette_run_suffix	TB	cMax = (PrefixOffset << 1) > PaletteMaxRunMinus1 ? (PaletteMaxRunMinus1 − PrefixOffset) : (PrefixOffset − 1)
	palette_escape_val	9.3.3.12	cIdx, cu_transquant_bypass_flag
delta_qp( )	cu_qp_delta_abs	9.3.3.10	-
	cu_qp_delta_sign_flag	FL	cMax = 1
chroma_qp_offset( )	cu_chroma_qp_offset_flag	FL	cMax = 1
	cu_chroma_qp_offset_idx	TR	cMax = chroma_qp_offset_list_len_minus1, cRiceParam = 0

9.3.3.2 Truncated Rice binarization process

Input to this process is a request for a truncated Rice (TR) binarization, cMax and cRiceParam.

Output of this process is the TR binarization associating each value symbolVal with a corresponding bin string.

A TR bin string is a concatenation of a prefix bin string and, when present, a suffix bin string.

For the derivation of the prefix bin string, the following applies:

– The prefix value of symbolVal, prefixVal, is derived as follows:

$$\text{prefixVal} = \text{symbolVal} \;>>\; \text{cRiceParam} \tag{9-11}$$

– The prefix of the TR bin string is specified as follows:
  – If prefixVal is less than cMax >> cRiceParam, the prefix bin string is a bit string of length prefixVal + 1 indexed by binIdx. The bins for binIdx less than prefixVal are equal to 1. The bin with binIdx equal to prefixVal is equal to 0. Table 9-44 illustrates the bin strings of this unary binarization for prefixVal.
  – Otherwise, the bin string is a bit string of length cMax >> cRiceParam with all bins being equal to 1.

Table 9-44 – Bin string of the unary binarization (informative)

prefixVal	Bin string					
0	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
...						
binIdx	0	1	2	3	4	5

When cMax is greater than symbolVal and cRiceParam is greater than 0, the suffix of the TR bin string is present and it is derived as follows:

– The suffix value suffixVal is derived as follows:

$$\text{suffixVal} = \text{symbolVal} - (\,(\,\text{prefixVal}\,) \;<<\; \text{cRiceParam}\,) \tag{9-12}$$

– The suffix of the TR bin string is specified by invoking the fixed-length (FL) binarization process as specified in clause 9.3.3.5 for suffixVal with a cMax value equal to (1 << cRiceParam) − 1.

NOTE – For the input parameter cRiceParam = 0, the TR binarization is exactly a truncated unary binarization and it is always invoked with a cMax value equal to the largest possible value of the syntax element being decoded.

9.3.3.3 k-th order Exp-Golomb binarization process

Inputs to this process is a request for a k-th order Exp-Golomb (EGk) binarization.

Output of this process is the EGk binarization associating each value symbolVal with a corresponding bin string.

The bin string of the EGk binarization process for each value symbolVal is specified as follows, where each call of the function put(X), with X being equal to 0 or 1, adds the binary value X at the end of the bin string:

```
absV = Abs( symbolVal )
stopLoop = 0
do
```

```
        if( absV  >=  ( 1  <<  k ) ) {
            put( 1 )
            absV = absV − ( 1  <<  k )
            k++
        } else {
            put( 0 )                                                    (9-13)
            while( k− − )
                put( ( absV  >>  k ) & 1 )
            stopLoop = 1
        }
    while( !stopLoop )
```

NOTE – The specification for the k-th order Exp-Golomb (EGk) code uses 1's and 0's in reverse meaning for the unary part of the Exp-Golomb code of 0-th order as specified in clause 9.2.

9.3.3.4 Limited EGk binarization process

This process is only invoked when extended_precision_processing_flag is equal to 1.

Inputs to this process is a request for a limited EGk binarization, the Rice parameter riceParam and the colour component cIdx.

Output of this process is the limited EGk binarization associating each value symbolVal with a corresponding bin string.

The variables log2TransformRange and maxPrefixExtensionLength are derived as follows:

$$\text{log2TransformRange} = \text{cIdx} == 0\ ?\ \text{Max}(\ 15, \text{BitDepth}_Y + 6\ ) : \text{Max}(\ 15, \text{BitDepth}_C + 6\ ) \quad (9\text{-}14)$$

$$\text{maxPrefixExtensionLength} = 28 - \text{log2TransformRange} \quad (9\text{-}15)$$

The bin string of the limited EGk binarization process for each value symbolVal is specified as follows, where each call of the function put(X), with X being equal to 0 or 1, adds the binary value X at the end of the bin string:

```
codeValue = symbolVal  >>  riceParam
PrefixExtensionLength = 0
while( ( PrefixExtensionLength < maxPrefixExtensionLength )  &&
        ( codeValue > ( ( 2  <<  PrefixExtensionLength ) − 2 ) ) ) {
    PrefixExtensionLength++
    put( 1 )
}
if( PrefixExtensionLength  = =  maxPrefixExtensionLength )
    escapeLength = log2TransformRange
else {                                                                  (9-16)
    escapeLength = PrefixExtensionLength + riceParam
    put( 0 )
}
symbolVal = symbolVal − ( ( ( 1  <<  PrefixExtensionLength ) − 1 )  <<  riceParam )
while( ( escapeLength− − ) > 0 )
    put( ( symbolVal  >>  escapeLength ) & 1 )
```

9.3.3.5 Fixed-length binarization process

Inputs to this process are a request for a fixed-length (FL) binarization and cMax.

Output of this process is the FL binarization associating each value symbolVal with a corresponding bin string.

FL binarization is constructed by using the fixedLength-bit unsigned integer bin string of the symbol value symbolVal, where fixedLength = Ceil(Log2(cMax + 1)). The indexing of bins for the FL binarization is such that the binIdx = 0 relates to the most significant bit with increasing values of binIdx towards the least significant bit.

9.3.3.6 Truncated Binary (TB) binarization process

Input to this process is a request for a TB binarization for a syntax element with value synVal and cMax. Output of this process is the TB binarization of the syntax element.The bin string of the TB binarization process of a syntax element synVal is specified as follows:

$$\begin{aligned} &n = cMax + 1 \\ &k = Floor(Log2(n)) \\ &u = (1 \ll (k + 1)) - n \end{aligned} \quad (9\text{-}17)$$

– If synVal is less than u, the TB bin string is derived by invoking the FL binarization process specified in clause 9.3.3.5 for synVal with a cMax value equal to (1 << k) − 1.

– Otherwise (synVal is greater than or equal to u), the TB bin string is derived by invoking the FL binarization process specified in clause 9.3.3.5 for (synVal + u) with a cMax value equal to (1 << (k + 1)) − 1.

9.3.3.7 Binarization process for part_mode

Inputs to this process are a request for a binarization for the syntax element part_mode, a luma location (xCb, yCb), specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture and a variable log2CbSize specifying the current luma coding block size.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element part_mode is specified in Table 9-45 depending on the values of CuPredMode[xCb][yCb] and log2CbSize.

Table 9-45 – Binarization for part_mode

CuPredMode[xCb][yCb]	**part_mode**	**PartMode**	**Bin string**			
			log2CbSize > MinCbLog2SizeY		log2CbSize = = MinCbLog2SizeY	
			!amp_enabled_flag	amp_enabled_flag	log2CbSize = = 3	log2CbSize > 3
MODE_INTRA	0	PART_2Nx2N	-	-	1	1
	1	PART_NxN	-	-	0	0
MODE_INTER	0	PART_2Nx2N	1	1	1	1
	1	PART_2NxN	01	011	01	01
	2	PART_Nx2N	00	001	00	001
	3	PART_NxN	-	-	-	000
	4	PART_2NxnU	-	0100	-	-
	5	PART_2NxnD	-	0101	-	-
	6	PART_nLx2N	-	0000	-	-
	7	PART_nRx2N	-	0001	-	-

9.3.3.8 Binarization process for intra_chroma_pred_mode

Input to this process is a request for a binarization for the syntax element intra_chroma_pred_mode.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element intra_chroma_pred_mode is specified in Table 9-46.

Table 9-46 – Binarization for intra_chroma_pred_mode

Value of intra_chroma_pred_mode	Bin string
4	0
0	100
1	101
2	110
3	111

9.3.3.9 Binarization process for inter_pred_idc

Inputs to this process are a request for a binarization for the syntax element inter_pred_idc, the current luma prediction block width nPbW and the current luma prediction block height nPbH.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element inter_pred_idc is specified in Table 9-47.

Table 9-47 – Binarization for inter_pred_idc

Value of inter_pred_idc	Name of inter_pred_idc	Bin string	
		(nPbW + nPbH) != 12	(nPbW + nPbH) = = 12
0	PRED_L0	00	0
1	PRED_L1	01	1
2	PRED_BI	1	-

9.3.3.10 Binarization process for cu_qp_delta_abs

Input to this process is a request for a binarization for the syntax element cu_qp_delta_abs.

Output of this process is the binarization of the syntax element.

The binarization of the syntax element cu_qp_delta_abs is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

– The prefix value of cu_qp_delta_abs, prefixVal, is derived as follows:

$$\text{prefixVal} = \text{Min}(\ \text{cu\_qp\_delta\_abs},\ 5\ ) \quad (9\text{-}18)$$

– The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with cMax = 5 and cRiceParam = 0.

When prefixVal is greater than 4, the suffix bin string is present and it is derived as follows:

– The suffix value of cu_qp_delta_abs, suffixVal, is derived as follows:

$$\text{suffixVal} = \text{cu\_qp\_delta\_abs} - 5 \quad (9\text{-}19)$$

– The suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for suffixVal with the Exp-Golomb order k set equal to 0.

9.3.3.11 Binarization process for coeff_abs_level_remaining[ ]

Input to this process is a request for a binarization for the syntax element coeff_abs_level_remaining[n], the current sub-block scan index i, baseLevel, the colour component cIdx and the luma location (x0, y0) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the picture.

Output of this process is the binarization of the syntax element.

Depending on the value of persistent_rice_adaptation_enabled_flag, the following applies:

– If persistent_rice_adaptation_enabled_flag is equal to 0, the variable initRiceValue is set equal to 0.

– Otherwise (persistent_rice_adaptation_enabled_flag is equal to 1), the following applies:

  – The variable sbType is derived as follows:

    o If transform_skip_flag[x0][y0][cIdx] is equal to 0 and cu_transquant_bypass_flag is equal to 0, the following applies:

$$\text{sbType} = 2 * (\text{cIdx} == 0 ~?~ 1 : 0) \quad (9\text{-}20)$$

    o Otherwise, the following applies:

$$\text{sbType} = 2 * (\text{cIdx} == 0 ~?~ 1 : 0) + 1 \quad (9\text{-}21)$$

  – The variable initRiceValue is derived as follows:

$$\text{initRiceValue} = \text{StatCoeff}[\text{sbType}] / 4 \quad (9\text{-}22)$$

  – If this process is invoked for the first time for the current sub-block scan index i, StatCoeff[sbType] is modified as follows:

```
if( coeff_abs_level_remaining[ n ] >= ( 3 << ( StatCoeff[ sbType ] / 4 ) ) )
    StatCoeff[ sbType ]++
else if( 2 * coeff_abs_level_remaining[ n ] < ( 1 << ( StatCoeff[ sbType ] / 4 ) ) &&
    StatCoeff[ sbType ] > 0 )
        StatCoeff[ sbType ]− −                                          (9-23)
```

The variables cLastAbsLevel and cLastRiceParam are derived as follows:

– If this process is invoked for the first time for the current sub-block scan index i, cLastAbsLevel is set equal to 0 and cLastRiceParam is set equal to initRiceValue.

– Otherwise (this process is not invoked for the first time for the current sub-block scan index i), cLastAbsLevel and cLastRiceParam are set equal to the values of cAbsLevel and cRiceParam, respectively, that have been derived during the last invocation of the binarization process for the syntax element coeff_abs_level_remaining[n] as specified in this clause.

The variable cAbsLevel is set equal to baseLevel + coeff_abs_level_remaining[n].

The variable cRiceParam is derived from cLastAbsLevel and cLastRiceParam as follows:

– If persistent_rice_adaptation_enabled_flag is equal to 0, the following applies:

$$\text{cRiceParam} = \text{Min}(\text{cLastRiceParam} + (\text{cLastAbsLevel} > (3 * (1 << \text{cLastRiceParam})) ~?~ 1 : 0), 4) \quad (9\text{-}24)$$

– Otherwise (persistent_rice_adaptation_enabled_flag is equal to 1), the following applies:

$$\text{cRiceParam} = \text{cLastRiceParam} + (\text{cLastAbsLevel} > (3 * (1 << \text{cLastRiceParam})) ~?~ 1 : 0) \quad (9\text{-}25)$$

The variable cMax is derived from cRiceParam as:

$$\text{cMax} = 4 << \text{cRiceParam} \quad (9\text{-}26)$$

The binarization of the syntax element coeff_abs_level_remaining[n] is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

– The prefix value of coeff_abs_level_remaining[n], prefixVal, is derived as follows:

$$\text{prefixVal} = \text{Min}(\text{cMax}, \text{coeff\_abs\_level\_remaining}[n]) \quad (9\text{-}27)$$

– The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with the variables cMax and cRiceParam as inputs.

When the prefix bin string is equal to the bit string of length 4 with all bits equal to 1, the suffix bin string is present and it is derived as follows:

– The suffix value of coeff_abs_level_remaining[n], suffixVal, is derived as follows:

$$\text{suffixVal} = \text{coeff\_abs\_level\_remaining}[\ n\ ] - \text{cMax} \tag{9-28}$$

– If extended_precision_processing_flag is equal to 0, the suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for the binarization of suffixVal with the Exp-Golomb order k set equal to cRiceParam + 1.

– Otherwise (extended_precision_processing_flag is equal to 1), the suffix bin string is specified by invoking the limited k-th order EGk binarization process as specified in clause 9.3.3.4 for the binarization of suffixVal with the variable riceParam set equal to cRiceParam + 1 and the colour component cIdx.

9.3.3.12 Binarization process for palette_escape_val

Input to this process is a request for a binarization for the syntax element palette_escape_val, cu_transquant_bypass_flag and colour component index cIdx.

Output of this process is the binarization of palette_escape_val.

The variable bitDepth is derived as follows:

$$\text{bitDepth} = (\ \text{cIdx}\ ==\ 0\ )\ ?\ \text{BitDepth}_Y : \text{BitDepth}_C \tag{9-29}$$

The binarization of palette_escape_val is derived as follows:

– If cu_transquant_bypass_flag is equal to 1, the binarization of palette_escape_val is derived by invoking the FL binarization process specified in clause 9.3.3.5 with the input parameter set to (1 << bitdepth) − 1.

Otherwise (cu_transquant_bypass_flag is equal to 0), the binarization of palette_escape_val is derived by invoking the k-th order Exp-Golomb binarization process specified in clause 9.3.3.3 with k set equal to 3.

9.3.3.13 Binarization process for palette_idx_idc

Input to this process is a request for a binarization for the syntax element palette_idx_idc and the variable MaxPaletteIndex.

Output of this process is the binarization of the syntax element.

The variable cMax is derived as follows:

– If this process is invoked for the first time for the current block, cMax is set equal to MaxPaletteIndex.

– Otherwise (this process is not invoked for the first time for the current block), cMax is set equal to MaxPaletteIndex minus 1.

The binarization for the palette_idx_idc is derived by invoking the TB binarization process specified in clause 9.3.3.6 with cMax.

9.3.3.14 Binarization process for num_palette_indices_minus1

Input to this process is a request for a binarization for the syntax element num_palette_indices_minus1, and MaxPaletteIndex.

Output of this process is the binarization of the syntax element.

The variable cRiceParam is derived as follows:

$$\text{cRiceParam} = 3 + (\ (\ \text{MaxPaletteIndex} + 1\ )\ >>\ 3\ ) \tag{9-30}$$

The variable cMax is derived from cRiceParam as:

$$\text{cMax} = 4\ <<\ \text{cRiceParam} \tag{9-31}$$

The binarization of the syntax element num_palette_indices_minus1 is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

– The prefix value of num_palette_indices_minus1, prefixVal, is derived as follows:

$$\text{prefixVal} = \text{Min}(\ \text{cMax}, \text{num\_palette\_indices\_minus1}\ ) \tag{9-32}$$

– The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with the variables cMax and cRiceParam as inputs.

When the prefix bin string is equal to the bit string of length 4 with all bits equal to 1, the suffix bin string is present and it is derived as follows:

– The suffix value of num_palette_indices_minus1, suffixVal, is derived as follows:

$$\text{suffixVal} = \text{num\_palette\_indices\_minus1} - \text{cMax} \tag{9-33}$$

The suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for the binarization of suffixVal with the Exp-Golomb order k set equal to cRiceParam + 1.

9.3.4 Decoding process flow

9.3.4.1 General

Inputs to this process are all bin strings of the binarization of the requested syntax element as specified in clause 9.3.3.

Output of this process is the value of the syntax element.

This process specifies how each bin of a bin string is parsed for each syntax element. After parsing each bin, the resulting bin string is compared to all bin strings of the binarization of the syntax element and the following applies:

– If the bin string is equal to one of the bin strings, the corresponding value of the syntax element is the output.

– Otherwise (the bin string is not equal to one of the bin strings), the next bit is parsed.

While parsing each bin, the variable binIdx is incremented by 1 starting with binIdx being set equal to 0 for the first bin.

The parsing of each bin is performed by invoking the derivation process for ctxTable, ctxIdx, and bypassFlag as specified in clause 9.3.4.2 with binIdx as input and ctxTable, ctxIdx and bypassFlag as outputs.

NOTE – As a consequence of invoking the process specified in clause 9.3.4.2, the arithmetic decoding process as specified in clause 9.3.4.3 is invoked with ctxTable, ctxIdx and bypassFlag as inputs and the value of the bin as output.

9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag

9.3.4.2.1 General

Input to this process is the position of the current bin within the bin string, binIdx.

Outputs of this process are ctxTable, ctxIdx and bypassFlag.

The values of ctxTable, ctxIdx and bypassFlag are derived as follows based on the entries for binIdx of the corresponding syntax element in Table 9-48:

– If the entry in Table 9-48 is not equal to "bypass", "terminate" or "na", the values of binIdx are decoded by invoking the DecodeDecision process as specified in clause 9.3.4.3.2 and the following applies:

  – ctxTable is specified in Table 9-4.

  – The variable ctxInc is specified by the corresponding entry in Table 9-48 and when more than one value is listed in Table 9-48 for a binIdx, the assignment process for ctxInc for that binIdx is further specified in the clauses given in parenthesis.

  – The variable ctxIdxOffset is specified by the lowest value of ctxIdx in Table 9-4 depending on the current value of initType.

  – ctxIdx is set equal to the sum of ctxInc and ctxIdxOffset.

  – bypassFlag is set equal to 0.

– Otherwise, if the entry in Table 9-48 is equal to "bypass", the values of binIdx are decoded by invoking the DecodeBypass process as specified in clause 9.3.4.3.4 and the following applies:

  – ctxTable is set equal to 0.

- ctxIdx is set equal to 0.
- bypassFlag is set equal to 1.

– Otherwise, if the entry in Table 9-48 is equal to "terminate", the values of binIdx are decoded by invoking the DecodeTerminate process as specified in clause 9.3.4.3.5 and the following applies:

- ctxTable is set equal to 0.
- ctxIdx is set equal to 0.
- bypassFlag is set equal to 0.

– Otherwise (the entry in Table 9-48 is equal to "na"), the values of binIdx do not occur for the corresponding syntax element.

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	**binIdx**					
	0	**1**	**2**	**3**	**4**	**>= 5**
end_of_slice_segment_flag	terminate	na	na	na	na	na
end_of_subset_one_bit	terminate	na	na	na	na	na
sao_merge_left_flag	0	na	na	na	na	na
sao_merge_up_flag	0	na	na	na	na	na
sao_type_idx_luma	0	bypass	na	na	na	na
sao_type_idx_chroma	0	bypass	na	na	na	na
sao_offset_abs[ ][ ][ ][ ]	bypass	bypass	bypass	bypass	bypass	na
sao_offset_sign[ ][ ][ ][ ]	bypass	na	na	na	na	na
sao_band_position[ ][ ][ ]	bypass	bypass	bypass	bypass	bypass	bypass
sao_eo_class_luma	bypass	bypass	na	na	na	na
sao_eo_class_chroma	bypass	bypass	na	na	na	na
split_cu_flag[ ][ ]	0,1,2 (clause 9.3.4.2.2)	na	na	na	na	na
cu_transquant_bypass_flag	0	na	na	na	na	na
cu_skip_flag	0,1,2 (clause 9.3.4.2.2)	na	na	na	na	na
palette_mode_flag	0	na	na	na	na	na
pred_mode_flag	0	na	na	na	na	na
part_mode log2CbSize = = MinCbLog2SizeY	0	1	2	bypass	na	na
part_mode log2CbSize > MinCbLog2SizeY	0	1	3	bypass	na	na
pcm_flag[ ][ ]	terminate	na	na	na	na	na
prev_intra_luma_pred_flag[ ][ ]	0	na	na	na	na	na
mpm_idx[ ][ ]	bypass	bypass	na	na	na	na
rem_intra_luma_pred_mode[ ][ ]	bypass	bypass	bypass	bypass	bypass	na
intra_chroma_pred_mode[ ][ ]	0	bypass	bypass	na	na	na
rqt_root_cbf	0	na	na	na	na	na
tu_residual_act_flag	0	na	na	na	na	na
merge_flag[ ][ ]	0	na	na	na	na	na
merge_idx[ ][ ]	0	bypass	bypass	bypass	na	na
inter_pred_idc[x0][y0]	(nPbW + nPbH) != 12 ? CtDepth[x0][y0] : 4	4	na	na	na	na

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	**1**	**2**	**3**	**4**	**>= 5**
ref_idx_l0[ ][ ]	0	1	bypass	bypass	bypass	bypass
ref_idx_l1[ ][ ]	0	1	bypass	bypass	bypass	bypass
mvp_l0_flag[ ][ ]	0	na	na	na	na	na
mvp_l1_flag[ ][ ]	0	na	na	na	na	na
split_transform_flag[ ][ ][ ]	5 − log2TrafoSize	na	na	na	na	na
cbf_cb[ ][ ][ ]	trafoDepth	na	na	na	na	na
cbf_cr[ ][ ][ ]	trafoDepth	na	na	na	na	na
cbf_luma[ ][ ][ ]	trafoDepth = = 0 ? 1 : 0	na	na	na	na	na
abs_mvd_greater0_flag[ ]	0	na	na	na	na	na
abs_mvd_greater1_flag[ ]	0	na	na	na	na	na
abs_mvd_minus2[ ]	bypass	bypass	bypass	bypass	bypass	bypass
mvd_sign_flag[ ]	bypass	na	na	na	na	na
cu_qp_delta_abs	0	1	1	1	1	bypass
cu_qp_delta_sign_flag	bypass	na	na	na	na	na
cu_chroma_qp_offset_flag	0	na	na	na	na	na
cu_chroma_qp_offset_idx	0	0	0	0	0	na
log2_res_scale_abs_plus1[c]	4 * c + 0	4 * c + 1	4 * c + 2	4 * c + 3	na	na
res_scale_sign_flag[c]	c	na	na	na	na	na
transform_skip_flag[ ][ ][ ]	0	na	na	na	na	na
explicit_rdpcm_flag[ ][ ][ ]	0	na	na	na	na	na
explicit_rdpcm_dir_flag[ ][ ][ ]	0	na	na	na	na	na
last_sig_coeff_x_prefix	0..17 (clause 9.3.4.2.3)					
last_sig_coeff_y_prefix	0..17 (clause 9.3.4.2.3)					
last_sig_coeff_x_suffix	bypass	bypass	bypass	na	na	na
last_sig_coeff_y_suffix	bypass	bypass	bypass	na	na	na
coded_sub_block_flag[ ][ ]	0..3 (clause 9.3.4.2.4)	na	na	na	na	na
sig_coeff_flag[ ][ ]	0..43 (clause 9.3.4.2.5)	na	na	na	na	na
coeff_abs_level_greater1_flag[ ]	0..23 (clause 9.3.4.2.6)	na	na	na	na	na
coeff_abs_level_greater2_flag[ ]	0..5 (clause 9.3.4.2.7)	na	na	na	na	na
coeff_abs_level_remaining[ ]	bypass	bypass	bypass	bypass	bypass	bypass
coeff_sign_flag[ ]	bypass	na	na	na	na	na
palette_predictor_run	bypass	bypass	bypass	bypass	bypass	bypass
num_signalled_palette_entries	bypass	bypass	bypass	bypass	bypass	bypass
new_palette_entries	bypass	bypass	bypass	bypass	bypass	bypass
palette_escape_val_present_flag	bypass	na	na	na	na	na
palette_transpose_flag	0	na	na	na	na	na
num_palette_indices_minus1	bypass	bypass	bypass	bypass	bypass	bypass

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	**1**	**2**	**3**	**4**	**>= 5**
palette_idx_idc	bypass	bypass	bypass	bypass	bypass	bypass
copy_above_palette_indices_flag and copy_above_indices_for_final_run_flag	0	na	na	na	na	na
palette_run_prefix	0..7 (clause 9.3.4.2.8)					
palette_run_suffix	bypass	bypass	bypass	bypass	bypass	bypass
palette_escape_val	bypass	bypass	bypass	bypass	bypass	bypass

9.3.4.2.2 Derivation process of ctxInc using left and above syntax elements

Input to this process is the luma location (x0, y0) specifying the top-left luma sample of the current luma block relative to the top-left sample of the current picture.

Output of this process is ctxInc.

The location (xNbL, yNbL) is set equal to (x0 − 1, y0) and the variable availableL, specifying the availability of the block located directly to the left of the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbL, yNbL) as inputs, and the output is assigned to availableL.

The location (xNbA, yNbA) is set equal to (x0, y0 − 1) and the variable availableA specifying the availability of the coding block located directly above the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbA, yNbA) as inputs, and the output is assigned to availableA.

The assignment of ctxInc for the syntax elements split_cu_flag[x0][y0] and cu_skip_flag[x0][y0] is specified in Table 9-49.

Table 9-49 – Specification of ctxInc using left and above syntax elements

Syntax element	condL	condA	ctxInc
split_cu_flag[x0][y0]	CtDepth[xNbL][yNbL] > cqtDepth	CtDepth[xNbA][yNbA] > cqtDepth	(condL && availableL) + (condA && availableA)
cu_skip_flag[x0][y0]	cu_skip_flag[xNbL][yNbL]	cu_skip_flag[xNbA][yNbA]	(condL && availableL) + (condA && availableA)

9.3.4.2.3 Derivation process of ctxInc for the syntax elements last_sig_coeff_x_prefix and last_sig_coeff_y_prefix

Inputs to this process are the variable binIdx, the colour component index cIdx and the transform block size log2TrafoSize.

Output of this process is the variable ctxInc.

The variables ctxOffset and ctxShift are derived as follows:

– If cIdx is equal to 0, ctxOffset is set equal to 3 * (log2TrafoSize − 2) + ((log2TrafoSize − 1) >> 2) and ctxShift is set equal to (log2TrafoSize + 1) >> 2.

– Otherwise (cIdx is greater than 0), ctxOffset is set equal to 15 and ctxShift is set equal to log2TrafoSize − 2.

The variable ctxInc is derived as follows:

$$\text{ctxInc} = (\text{binIdx} >> \text{ctxShift}) + \text{ctxOffset} \qquad (9\text{-}34)$$

9.3.4.2.4 Derivation process of ctxInc for the syntax element coded_sub_block_flag

Inputs to this process are the colour component index cIdx, the current sub-block scan location (xS, yS), the previously decoded bins of the syntax element coded_sub_block_flag and the transform block size log2TrafoSize.

Output of this process is the variable ctxInc.

The variable csbfCtx is derived using the current location (xS, yS), two previously decoded bins of the syntax element coded_sub_block_flag in scan order and the transform block size log2TrafoSize, as follows:

– csbfCtx is initialized with 0 as follows:

$$\text{csbfCtx} = 0 \tag{9-35}$$

– When xS is less than (1 << (log2TrafoSize − 2)) − 1, csbfCtx is modified as follows:

$$\text{csbfCtx} \mathrel{+}= \text{coded\_sub\_block\_flag}[\ xS + 1\ ][\ yS\ ] \tag{9-36}$$

– When yS is less than (1 << (log2TrafoSize − 2)) − 1, csbfCtx is modified as follows:

$$\text{csbfCtx} \mathrel{+}= \text{coded\_sub\_block\_flag}[\ xS\ ][\ yS + 1\ ] \tag{9-37}$$

The context index increment ctxInc is derived using the colour component index cIdx and csbfCtx as follows:

– If cIdx is equal to 0, ctxInc is derived as follows:

$$\text{ctxInc} = \text{Min}(\ \text{csbfCtx}, 1\ ) \tag{9-38}$$

– Otherwise (cIdx is greater than 0), ctxInc is derived as follows:

$$\text{ctxInc} = 2 + \text{Min}(\ \text{csbfCtx}, 1\ ) \tag{9-39}$$

9.3.4.2.5 Derivation process of ctxInc for the syntax element sig_coeff_flag

Inputs to this process are the colour component index cIdx, the luma location (x0, y0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, the current coefficient scan location (xC, yC), the scan order index scanIdx and the transform block size log2TrafoSize.

Output of this process is the variable ctxInc.

The variable sigCtx depends on the current location (xC, yC), the colour component index cIdx, the value of transform_skip_flag, the value of cu_transquant_bypass_flag, the transform block size and previously decoded bins of the syntax element coded_sub_block_flag. For the derivation of sigCtx, the following applies:

– If transform_skip_context_enabled_flag is equal to 1 and either or both transform_skip_flag[x0][y0][cIdx] is equal to 1 or cu_transquant_bypass_flag is equal to 1, sigCtx is derived as follows:

$$\text{sigCtx} = (\ \text{cIdx} == 0\ )\ ?\ 42 : 16 \tag{9-40}$$

– Otherwise, if log2TrafoSize is equal to 2, sigCtx is derived using ctxIdxMap[ ] specified in Table 9-50 as follows:

$$\text{sigCtx} = \text{ctxIdxMap}[\ (\ yC \ll 2\ ) + xC\ ] \tag{9-41}$$

– Otherwise, if xC + yC is equal to 0, sigCtx is derived as follows:

$$\text{sigCtx} = 0 \tag{9-42}$$

– Otherwise, sigCtx is derived using previous values of coded_sub_block_flag as follows:

  – The sub-block location (xS, yS) is set equal to (xC >> 2, yC >> 2).

  – The variable prevCsbf is set equal to 0.

  – When xS is less than (1 << (log2TrafoSize − 2)) − 1, the following applies:

$$\text{prevCsbf} \mathrel{+}= \text{coded\_sub\_block\_flag}[\ xS + 1\ ][\ yS\ ] \tag{9-43}$$

  – When yS is less than (1 << (log2TrafoSize − 2)) − 1, the following applies:

$$\text{prevCsbf} \mathrel{+}= (\ \text{coded\_sub\_block\_flag}[\ xS\ ][\ yS + 1\ ] \ll 1\ ) \tag{9-44}$$

  – The inner sub-block location (xP, yP) is set equal to (xC & 3, yC & 3).

– The variable sigCtx is derived as follows:

– If prevCsbf is equal to 0, the following applies:

$$\text{sigCtx} = (\text{xP} + \text{yP} == 0) ~?~ 2 : (\text{xP} + \text{yP} < 3) ~?~ 1 : 0 \quad (9\text{-}45)$$

– Otherwise, if prevCsbf is equal to 1, the following applies:

$$\text{sigCtx} = (\text{yP} == 0) ~?~ 2 : (\text{yP} == 1) ~?~ 1 : 0 \quad (9\text{-}46)$$

– Otherwise, if prevCsbf is equal to 2, the following applies:

$$\text{sigCtx} = (\text{xP} == 0) ~?~ 2 : (\text{xP} == 1) ~?~ 1 : 0 \quad (9\text{-}47)$$

– Otherwise (prevCsbf is equal to 3), the following applies:

$$\text{sigCtx} = 2 \quad (9\text{-}48)$$

– The variable sigCtx is modified as follows:

– If cIdx is equal to 0, the following applies:

– When (xS + yS) is greater than 0, the following applies:

$$\text{sigCtx} \mathrel{+}= 3 \quad (9\text{-}49)$$

– The variable sigCtx is modified as follows:

– If log2TrafoSize is equal to 3, the following applies:

$$\text{sigCtx} \mathrel{+}= (\text{scanIdx} == 0) ~?~ 9 : 15 \quad (9\text{-}50)$$

– Otherwise, the following applies:

$$\text{sigCtx} \mathrel{+}= 21 \quad (9\text{-}51)$$

– Otherwise (cIdx is greater than 0), the following applies:

– If log2TrafoSize is equal to 3, the following applies:

$$\text{sigCtx} \mathrel{+}= 9 \quad (9\text{-}52)$$

– Otherwise, the following applies:

$$\text{sigCtx} \mathrel{+}= 12 \quad (9\text{-}53)$$

The context index increment ctxInc is derived using the colour component index cIdx and sigCtx as follows:

– If cIdx is equal to 0, ctxInc is derived as follows:

$$\text{ctxInc} = \text{sigCtx} \quad (9\text{-}54)$$

– Otherwise (cIdx is greater than 0), ctxInc is derived as follows:

$$\text{ctxInc} = 27 + \text{sigCtx} \quad (9\text{-}55)$$

Table 9-50 – Specification of ctxIdxMap[i]

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
ctxIdxMap[i]	0	1	4	5	2	3	4	5	6	6	8	8	7	7	8

9.3.4.2.6 Derivation process of ctxInc for the syntax element coeff_abs_level_greater1_flag

Inputs to this process are the colour component index cIdx, the current sub-block scan index i and the current coefficient scan index n within the current sub-block.

Output of this process is the variable ctxInc.

The variable ctxSet specifies the current context set and for its derivation the following applies:

- If this process is invoked for the first time for the current sub-block scan index i, the following applies:
  - The variable ctxSet is initialized as follows:
    - If the current sub-block scan index i is equal to 0 or cIdx is greater than 0, the following applies:

$$\text{ctxSet} = 0 \tag{9-56}$$

    - Otherwise (i is greater than 0 and cIdx is equal to 0), the following applies:

$$\text{ctxSet} = 2 \tag{9-57}$$

  - The variable lastGreater1Ctx is derived as follows:
    - If the current sub-block with scan index i is the first one to be processed in this clause for the current transform block, the variable lastGreater1Ctx is set equal to 1.
    - Otherwise, the following applies:
      - The variable lastGreater1Ctx is set equal to the value of greater1Ctx that has been derived during the last invocation of the process specified in this clause for a previous sub-block.
      - When lastGreater1Ctx is greater than 0, the variable lastGreater1Flag is set equal to the value of the syntax element coeff_abs_level_greater1_flag that has been used during the last invocation of the process specified in this clause for a previous sub-block and lastGreater1Ctx is modified as follows:
        - If lastGreater1Flag is equal to 1, lastGreater1Ctx is set equal to 0.
        - Otherwise (lastGreater1Flag is equal to 0), lastGreater1Ctx is incremented by 1.
  - When lastGreater1Ctx is equal to 0, ctxSet is incremented by one as follows:

$$\text{ctxSet} = \text{ctxSet} + 1 \tag{9-58}$$

  - The variable greater1Ctx is set equal to 1.
- Otherwise (this process is not invoked for the first time for the current sub-block scan index i), the following applies:
  - The variable ctxSet is set equal to the variable ctxSet that has been derived during the last invocation of the process specified in this clause.
  - The variable greater1Ctx is set equal to the variable greater1Ctx that has been derived during the last invocation of the process specified in this clause.
  - When greater1Ctx is greater than 0, the variable lastGreater1Flag is set equal to the syntax element coeff_abs_level_greater1_flag that has been used during the last invocation of the process specified in this clause and greater1Ctx is modified as follows:
    - If lastGreater1Flag is equal to 1, greater1Ctx is set equal to 0.
    - Otherwise (lastGreater1Flag is equal to 0), greater1Ctx is incremented by 1.

The context index increment ctxInc is derived using the current context set ctxSet and the current context greater1Ctx as follows:

$$\text{ctxInc} = (\ \text{ctxSet} * 4\ ) + \text{Min}(\ 3, \text{greater1Ctx}\ ) \tag{9-59}$$

When cIdx is greater than 0, ctxInc is modified as follows:

$$\text{ctxInc} = \text{ctxInc} + 16 \tag{9-60}$$

9.3.4.2.7 Derivation process of ctxInc for the syntax element coeff_abs_level_greater2_flag

Inputs to this process are the colour component index cIdx, the current sub-block scan index i and the current coefficient scan index n within the current sub-block.

Output of this process is the variable ctxInc.

The variable ctxSet specifies the current context set and is set equal to the value of the variable ctxSet that has been derived in clause 9.3.4.2.6 for the same subset i.

The context index increment ctxInc is set equal to the variable ctxSet as follows:

$$\text{ctxInc} = \text{ctxSet} \qquad (9\text{-}61)$$

When cIdx is greater than 0, ctxInc is modified as follows:

$$\text{ctxInc} = \text{ctxInc} + 4 \qquad (9\text{-}62)$$

9.3.4.2.8 Derivation process of ctxInc for the syntax element palette_run_prefix

Inputs to this process are the bin index binIdx and the syntax elements copy_above_palette_indices_flag and palette_idx_idc.

Output of this process is the variable ctxInc.

The variable ctxInc is derived as follows:

– If copy_above_palette_indices_flag is equal to 0 and binIdx is equal to 0, ctxInc is derived as follows:

$$\text{ctxInc} = (\ \text{palette\_idx\_idc} < 1\ )\ ?\ 0 : (\ (\ \text{palette\_idx\_idc} < 3\ )\ ?\ 1 : 2\ ) \qquad (9\text{-}63)$$

– Otherwise, ctxInc is provided by Table 9-51.

Table 9-51 – Specification of ctxIdxMap[copy_above_palette_indices_flag][binIdx]

binIdx	0	1	2	3	4	> 4
copy_above_palette_indices_flag = = 1	5	6	6	7	7	bypass
copy_above_palette_indices_flag = = 0	0, 1, 2	3	3	4	4	bypass

9.3.4.3 Arithmetic decoding process

9.3.4.3.1 General

Inputs to this process are ctxTable, ctxIdx and bypassFlag, as derived in clause 9.3.4.2, and the state variables ivlCurrRange and ivlOffset of the arithmetic decoding engine.

Output of this process is the value of the bin.

Figure 9-5 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxTable and the ctxIdx are passed to the arithmetic decoding process DecodeBin(ctxTable, ctxIdx), which is specified as follows:

– If bypassFlag is equal to 1, DecodeBypass( ) as specified in clause 9.3.4.3.4 is invoked.

– Otherwise, if bypassFlag is equal to 0, ctxTable is equal to 0 and ctxIdx is equal to 0, DecodeTerminate( ) as specified in clause 9.3.4.3.5 is invoked.

– Otherwise (bypassFlag is equal to 0 and ctxTable is not equal to 0), DecodeDecision( ) as specified in clause 9.3.4.3.2 is invoked.

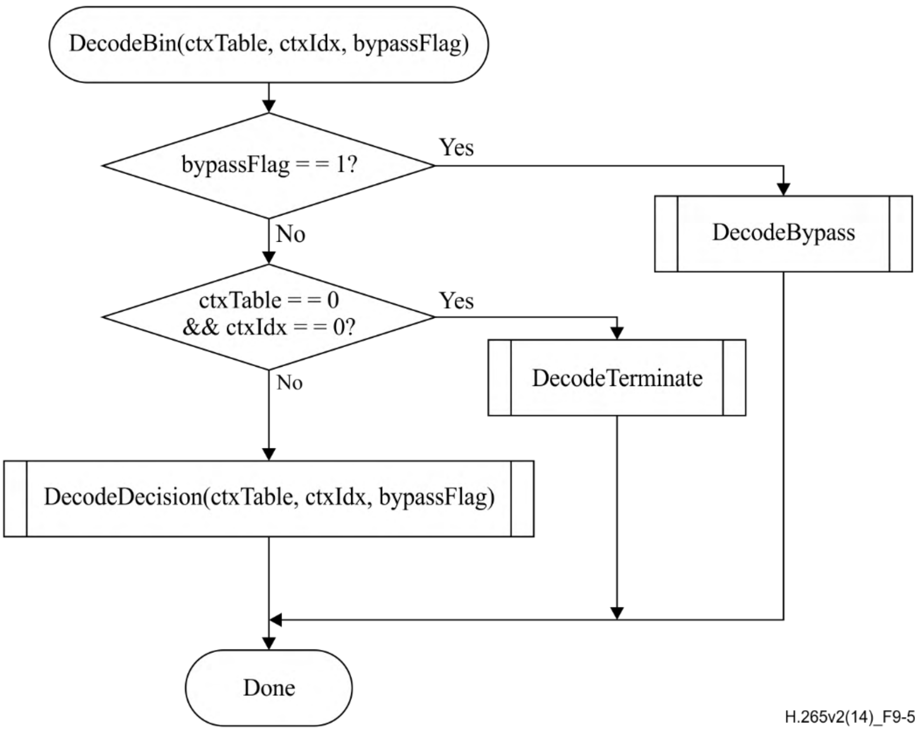


Figure 9-5 – Overview of the arithmetic decoding process for a single bin (informative)

NOTE – Arithmetic coding is based on the principle of recursive interval subdivision. Given a probability estimation p(0) and p(1) = 1 − p(0) of a binary decision (0, 1), an initially given code sub-interval with the range ivlCurrRange will be subdivided into two sub-intervals having range p(0) * ivlCurrRange and ivlCurrRange − p(0) * ivlCurrRange, respectively. Depending on the decision, which has been observed, the corresponding sub-interval will be chosen as the new code interval, and a binary code string pointing into that interval will represent the sequence of observed binary decisions. It is useful to distinguish between the most probable symbol (MPS) and the least probable symbol (LPS), so that binary decisions have to be identified as either MPS or LPS, rather than 0 or 1. Given this terminology, each context is specified by the probability p_{LPS} of the LPS and the value of MPS (valMps), which is either 0 or 1. The arithmetic core engine in this Specification has three distinct properties:

– The probability estimation is performed by means of a finite-state machine with a table-based transition process between 64 different representative probability states { p_{LPS}(pStateIdx) | 0 <= pStateIdx < 64 } for the LPS probability p_{LPS}. The numbering of the states is arranged in such a way that the probability state with index pStateIdx = 0 corresponds to an LPS probability value of 0.5, with decreasing LPS probability towards higher state indices.
– The range ivlCurrRange representing the state of the coding engine is quantized to a small set {Q_1,...,Q_4} of pre-set quantization values prior to the calculation of the new interval range. Storing a table containing all 64x4 pre-computed product values of Q_i * p_{LPS}(pStateIdx) allows a multiplication-free approximation of the product ivlCurrRange * p_{LPS}(pStateIdx).
– For syntax elements or parts thereof for which an approximately uniform probability distribution is assumed to be given a separate simplified encoding and decoding bypass process is used.

9.3.4.3.2 Arithmetic decoding process for a binary decision

9.3.4.3.2.1 General

Inputs to this process are the variables ctxTable, ctxIdx, ivlCurrRange and ivlOffset.

Outputs of this process are the decoded value binVal and the updated variables ivlCurrRange and ivlOffset.

Figure 9-6 shows the flowchart for decoding a single decision (DecodeDecision):

1. The value of the variable ivlLpsRange is derived as follows:
   – Given the current value of ivlCurrRange, the variable qRangeIdx is derived as follows:

     qRangeIdx =(ivlCurrRange >> 6) & 3 (9-64)

   – Given qRangeIdx and pStateIdx associated with ctxTable and ctxIdx, the value of the variable rangeTabLps as specified in Table 9-52 is assigned to ivlLpsRange:

     ivlLpsRange = rangeTabLps[pStateIdx][qRangeIdx] (9-65)

2. The variable ivlCurrRange is set equal to ivlCurrRange − ivlLpsRange and the following applies:

   – If ivlOffset is greater than or equal to ivlCurrRange, the variable binVal is set equal to 1 − valMps, ivlOffset is decremented by ivlCurrRange and ivlCurrRange is set equal to ivlLpsRange.

   – Otherwise, the variable binVal is set equal to valMps.

Given the value of binVal, the state transition is performed as specified in clause 9.3.4.3.2.2. Depending on the current value of ivlCurrRange, renormalization is performed as specified in clause 9.3.4.3.3.

9.3.4.3.2.2 State transition process

Inputs to this process are the current pStateIdx, the decoded value binVal and valMps values of the context variable associated with ctxTable and ctxIdx.

Outputs of this process are the updated pStateIdx and valMps of the context variable associated with ctxIdx.

Depending on the decoded value binVal, the update of the two variables pStateIdx and valMps associated with ctxIdx is derived as follows:

```
if( binVal  = =  valMps )
    pStateIdx = transIdxMps( pStateIdx )
else {                                                    (9-66)
    if( pStateIdx  = =  0 )
        valMps = 1 – valMps
    pStateIdx = transIdxLps( pStateIdx )
}
```

Table 9-53 specifies the transition rules transIdxMps( ) and transIdxLps( ) after decoding the value of valMps and 1 – valMps, respectively.

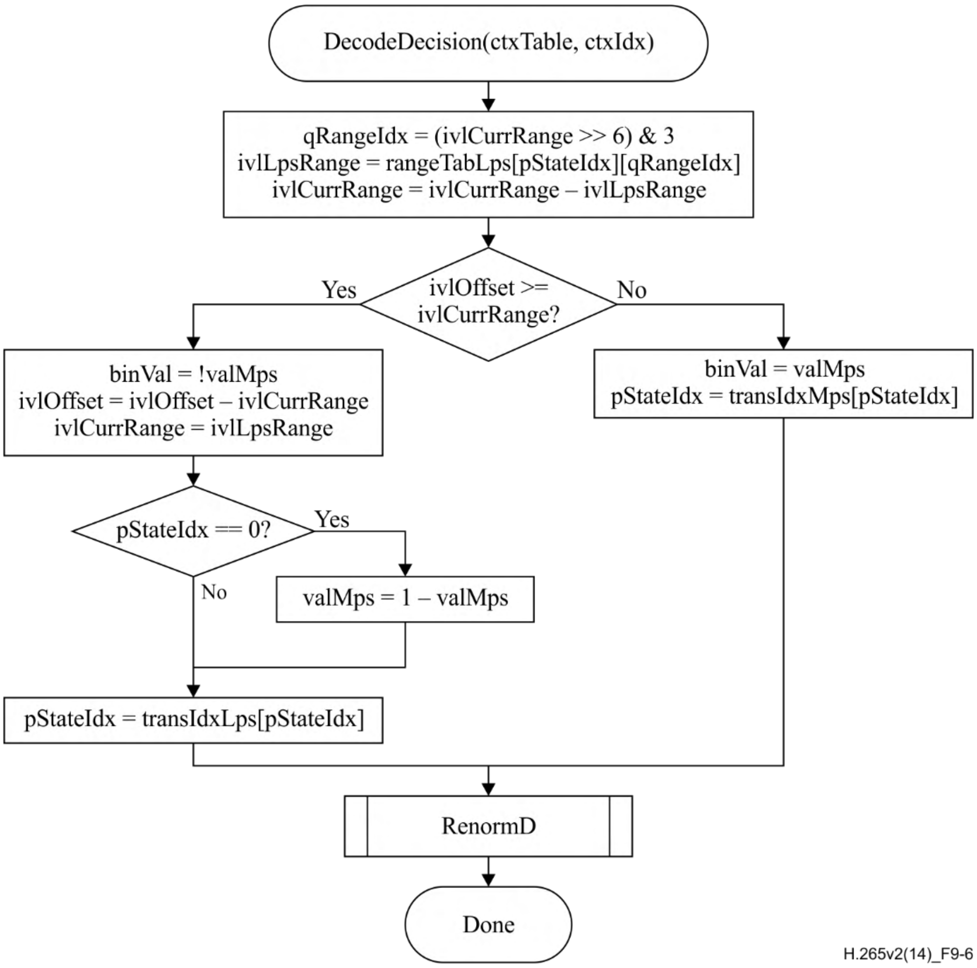


Figure 9-6 – Flowchart for decoding a decision

Table 9-52 – Specification of rangeTabLps depending on the values of pStateIdx and qRangeIdx

pStateIdx	qRangeIdx				pStateIdx	qRangeIdx			
	0	**1**	**2**	**3**		**0**	**1**	**2**	**3**
0	128	176	208	240	**32**	27	33	39	45
1	128	167	197	227	**33**	26	31	37	43
2	128	158	187	216	**34**	24	30	35	41
3	123	150	178	205	**35**	23	28	33	39
4	116	142	169	195	**36**	22	27	32	37
5	111	135	160	185	**37**	21	26	30	35
6	105	128	152	175	**38**	20	24	29	33
7	100	122	144	166	**39**	19	23	27	31
8	95	116	137	158	**40**	18	22	26	30
9	90	110	130	150	**41**	17	21	25	28
10	85	104	123	142	**42**	16	20	23	27
11	81	99	117	135	**43**	15	19	22	25
12	77	94	111	128	**44**	14	18	21	24
13	73	89	105	122	**45**	14	17	20	23
14	69	85	100	116	**46**	13	16	19	22
15	66	80	95	110	**47**	12	15	18	21
16	62	76	90	104	**48**	12	14	17	20
17	59	72	86	99	**49**	11	14	16	19
18	56	69	81	94	**50**	11	13	15	18
19	53	65	77	89	**51**	10	12	15	17
20	51	62	73	85	**52**	10	12	14	16
21	48	59	69	80	**53**	9	11	13	15
22	46	56	66	76	**54**	9	11	12	14
23	43	53	63	72	**55**	8	10	12	14
24	41	50	59	69	**56**	8	9	11	13
25	39	48	56	65	**57**	7	9	11	12
26	37	45	54	62	**58**	7	9	10	12
27	35	43	51	59	**59**	7	8	10	11
28	33	41	48	56	**60**	6	8	9	11
29	32	39	46	53	**61**	6	7	9	10
30	30	37	43	50	**62**	6	7	8	9
31	29	35	41	48	**63**	2	2	2	2

Table 9-53 – State transition table

pStateIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
transIdxLps	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
transIdxLps	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMps	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
transIdxLps	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMps	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
transIdxLps	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMps	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

9.3.4.3.3 Renormalization process in the arithmetic decoding engine

Inputs to this process are bits from slice segment data and the variables ivlCurrRange and ivlOffset.

Outputs of this process are the updated variables ivlCurrRange and ivlOffset.

A flowchart of the renormalization is shown in Figure 9-7. The current value of ivlCurrRange is first compared to 256 and then the following applies:

– If ivlCurrRange is greater than or equal to 256, no renormalization is needed and the RenormD process is finished;

– Otherwise (ivlCurrRange is less than 256), the renormalization loop is entered. Within this loop, the value of ivlCurrRange is doubled, i.e., left-shifted by 1 and a single bit is shifted into ivlOffset by using read_bits(1).

The bitstream shall not contain data that result in a value of ivlOffset being greater than or equal to ivlCurrRange upon completion of this process.

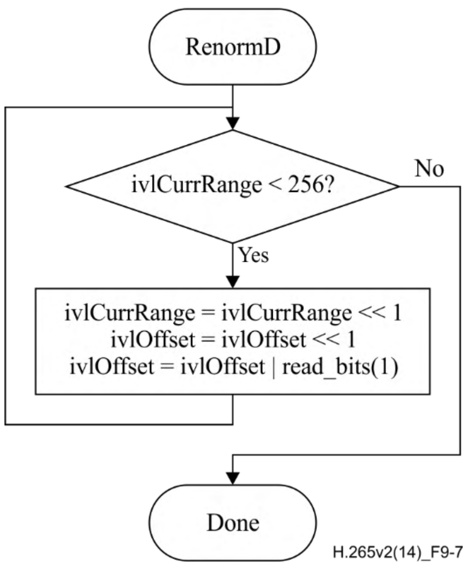


Figure 9-7 – Flowchart of renormalization

9.3.4.3.4 Bypass decoding process for binary decisions

Inputs to this process are bits from slice segment data and the variables ivlCurrRange and ivlOffset.

Outputs of this process are the updated variable ivlOffset and the decoded value binVal.

The bypass decoding process is invoked when bypassFlag is equal to 1. Figure 9-8 shows a flowchart of the corresponding process.

First, the value of ivlOffset is doubled, i.e., left-shifted by 1 and a single bit is shifted into ivlOffset by using read_bits(1). Then, the value of ivlOffset is compared to the value of ivlCurrRange and then the following applies:

– If ivlOffset is greater than or equal to ivlCurrRange, the variable binVal is set equal to 1 and ivlOffset is decremented by ivlCurrRange.

– Otherwise (ivlOffset is less than ivlCurrRange), the variable binVal is set equal to 0.

The bitstream shall not contain data that result in a value of ivlOffset being greater than or equal to ivlCurrRange upon completion of this process.

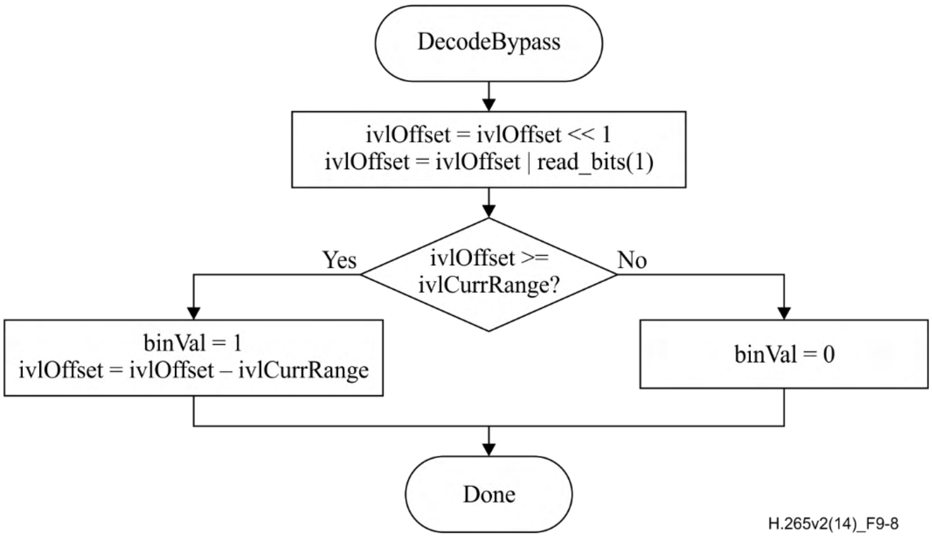


Figure 9-8 – Flowchart of bypass decoding process

9.3.4.3.5 Decoding process for binary decisions before termination

Inputs to this process are bits from slice segment data and the variables ivlCurrRange and ivlOffset.

Outputs of this process are the updated variables ivlCurrRange and ivlOffset, and the decoded value binVal.

This decoding process applies to decoding of end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag corresponding to ctxTable equal to 0 and ctxIdx equal to 0. Figure 9-9 shows the flowchart of the corresponding decoding process, which is specified as follows:

First, the value of ivlCurrRange is decremented by 2. Then, the value of ivlOffset is compared to the value of ivlCurrRange and then the following applies:

– If ivlOffset is greater than or equal to ivlCurrRange, the variable binVal is set equal to 1, no renormalization is carried out, and CABAC decoding is terminated. The last bit inserted in register ivlOffset is equal to 1. When decoding end_of_slice_segment_flag, this last bit inserted in register ivlOffset is interpreted as rbsp_stop_one_bit. When decoding end_of_subset_one_bit, this last bit inserted in register ivlOffset is interpreted as alignment_bit_equal_to_one.

– Otherwise (ivlOffset is less than ivlCurrRange), the variable binVal is set equal to 0 and renormalization is performed as specified in clause 9.3.4.3.3.

NOTE – This procedure may also be implemented using DecodeDecision(ctxTable, ctxIdx, bypassFlag) with ctxTable = 0, ctxIdx = 0 and bypassFlag = 0. In the case where the decoded value is equal to 1, seven more bits would be read by DecodeDecision(ctxTable, ctxIdx, bypassFlag) and a decoding process would have to adjust its bitstream pointer accordingly to properly decode following syntax elements.

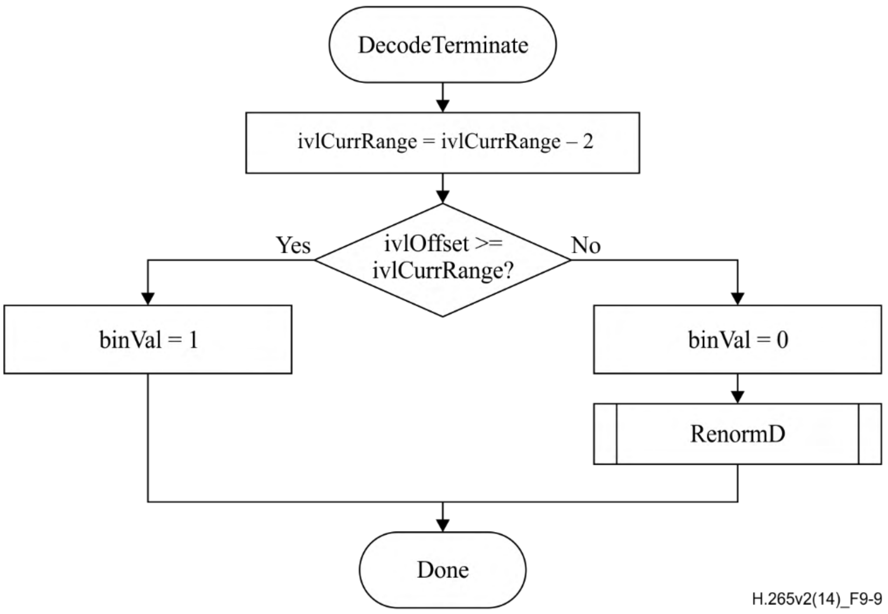


Figure 9-9 – Flowchart of decoding a decision before termination

9.3.4.3.6 Alignment process prior to aligned bypass decoding

Input to this process is the variable ivlCurrRange.

Output of this process is the updated variable ivlCurrRange.

This process applies prior to the decoding of syntax elements coeff_abs_level_remaining[ ] and coeff_sign_flag[ ].

ivlCurrRange is set equal to 256.

NOTE – When ivlCurrRange is 256, ivlOffset and the bit-stream can be considered as a shift register, and binVal as the register's second most significant bit (the most significant bit is always 0 due to the restriction of ivlOffset being less than ivlCurrRange).

9.3.5 Arithmetic encoding process (informative)

9.3.5.1 General

This clause does not form an integral part of this Specification.

Inputs to this process are decisions that are to be encoded and written.

Outputs of this process are bits that are written to the RBSP.

This informative clause describes an arithmetic encoding engine that matches the arithmetic decoding engine described in clause 9.3.4.3. The encoding engine is essentially symmetric with the decoding engine, i.e., procedures are called in the same order. The following procedures are described in this clause: InitEncoder, EncodeDecision, EncodeBypass, EncodeTerminate, which correspond to InitDecoder, DecodeDecision, DecodeBypass and DecodeTerminate, respectively. The state of the arithmetic encoding engine is represented by a value of the variable ivlLow pointing to the lower end of a sub-interval and a value of the variable ivlCurrRange specifying the corresponding range of that sub-interval.

9.3.5.2 Initialization process for the arithmetic encoding engine (informative)

This clause does not form an integral part of this Specification.

This process is invoked before encoding the first coding block of a slice segment, and after encoding any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data for a coding unit with pcm_flag equal to 1.

Outputs of this process are the values ivlLow, ivlCurrRange, firstBitFlag, bitsOutstanding and BinCountsInNalUnits of the arithmetic encoding engine.

In the initialization procedure of the encoder, ivlLow is set equal to 0 and ivlCurrRange is set equal to 510. Furthermore, firstBitFlag is set equal to 1 and the counter bitsOutstanding is set equal to 0.

Depending on whether the current slice segment is the first slice segment of a coded picture, the following applies:

– If the current slice segment is the first slice segment of a coded picture, the counter BinCountsInNalUnits is set equal to 0.

– Otherwise (the current slice segment is not the first slice segment of a coded picture), the counter BinCountsInNalUnits is not modified. The value of BinCountsInNalUnits is the result of encoding all the slice segments of a coded picture that precede the current slice segment in decoding order. After initializing for the first slice segment of a coded picture as specified in this clause, BinCountsInNalUnits is incremented as specified in clauses 9.3.5.3, 9.3.5.5 and 9.3.5.6.

NOTE – The minimum register precision required for storing the values of the variables ivlLow and ivlCurrRange after invocation of any of the arithmetic encoding processes specified in clauses 9.3.5.3, 9.3.5.5 and 9.3.5.6 is 10 bits and 9 bits, respectively. The encoding process for a binary decision (EncodeDecision) as specified in clause 9.3.5.3 and the encoding process for a binary decision before termination (EncodeTerminate) as specified in clause 9.3.5.6 require a minimum register precision of 10 bits for the variable ivlLow and a minimum register precision of 9 bits for the variable ivlCurrRange. The bypass encoding process for binary decisions (EncodeBypass) as specified in clause 9.3.5.5 requires a minimum register precision of 11 bits for the variable ivlLow and a minimum register precision of 9 bits for the variable ivlCurrRange. The precision required for the counters bitsOutstanding and BinCountsInNalUnits should be sufficiently large to prevent overflow of the related registers. When maxBinCountInSlice denotes the maximum total number of binary decisions to encode in one slice segment and maxBinCountInPic denotes the maximum total number of binary decisions to encode a picture, the minimum register precision required for the variables bitsOutstanding and BinCountsInNalUnits is given by Ceil(Log2(maxBinCountInSlice + 1)) and Ceil(Log2(maxBinCountInPic + 1)), respectively.

9.3.5.3 Encoding process for a binary decision (informative)

This clause does not form an integral part of this Specification.

Inputs to this process are the context index ctxIdx, the value of binVal to be encoded and the variables ivlCurrRange, ivlLow and BinCountsInNalUnits.

Outputs of this process are the variables ivlCurrRange, ivlLow and BinCountsInNalUnits.

Figure 9-10 shows the flowchart for encoding a single decision. In a first step, the variable ivlLpsRange is derived as follows:

Given the current value of ivlCurrRange, ivlCurrRange is mapped to the index qRangeIdx of a quantized value of ivlCurrRange by using Equation 9-64. The value of qRangeIdx and the value of pStateIdx associated with ctxIdx are used to determine the value of the variable rangeTabLps as specified in Table 9-52, which is assigned to ivlLpsRange. The value of ivlCurrRange − ivlLpsRange is assigned to ivlCurrRange.

In a second step, the value of binVal is compared to valMps associated with ctxIdx. When binVal is different from valMps, ivlCurrRange is added to ivlLow and ivlCurrRange is set equal to the value ivlLpsRange. Given the encoded decision, the state transition is performed as specified in clause 9.3.4.3.2.2. Depending on the current value of ivlCurrRange, renormalization is performed as specified in clause 9.3.5.4. Finally, the variable BinCountsInNalUnits is incremented by 1.

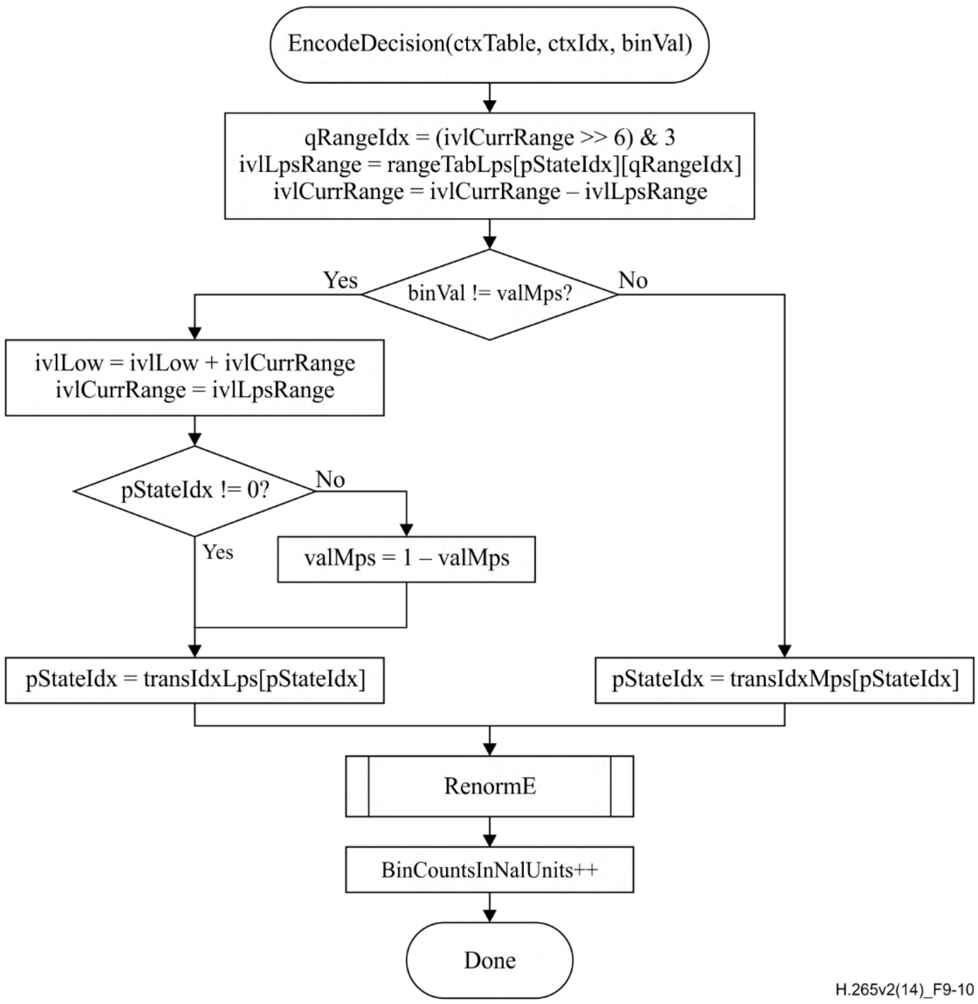


Figure 9-10 – Flowchart for encoding a decision

9.3.5.4 Renormalization process in the arithmetic encoding engine (informative)

This clause does not form an integral part of this Specification.

Inputs to this process are the variables ivlCurrRange, ivlLow, firstBitFlag and bitsOutstanding.

Outputs of this process are zero or more bits written to the RBSP and the updated variables ivlCurrRange, ivlLow, firstBitFlag and bitsOutstanding.

Renormalization in the encoder is illustrated in Figure 9-11.

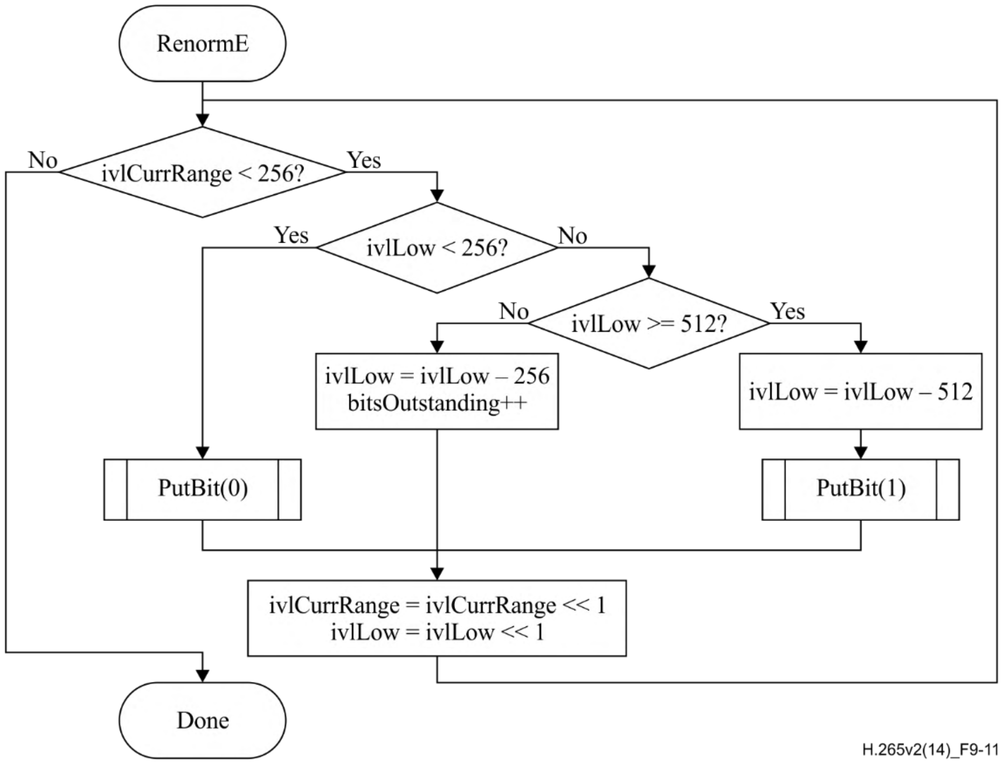


Figure 9-11 – Flowchart of renormalization in the encoder

The PutBit( ) procedure described in Figure 9-12 provides carry over control. It uses the function WriteBits(B, N) that writes N bits with value B to the bitstream and advances the bitstream pointer by N bit positions. This function assumes the existence of a bitstream pointer with an indication of the position of the next bit to be written to the bitstream by the encoding process.

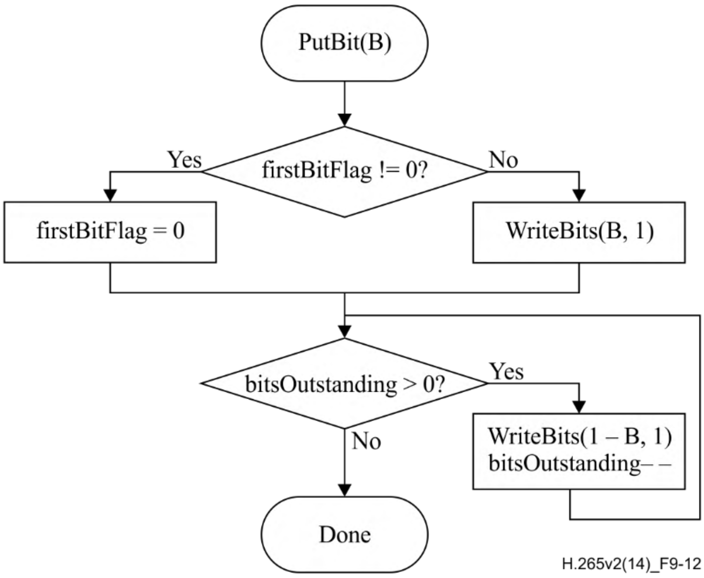


Figure 9-12 – Flowchart of PutBit(B)

9.3.5.5 Bypass encoding process for binary decisions (informative)

This clause does not form an integral part of this Specification.

Inputs to this process are the variables binVal, ivlLow, ivlCurrRange, bitsOutstanding and BinCountsInNalUnits.

Output of this process is a bit written to the RBSP and the updated variables ivlLow, bitsOutstanding and BinCountsInNalUnits.

This encoding process applies to all binary decisions with bypassFlag equal to 1.

When cabac_bypass_alignment_enabled_flag is equal to 1 and coeff_abs_level_remaining[ ] is present for any coefficients in the current sub-block, an alignment process is performed. This alignment process applies prior to the encoding of the syntax elements coeff_abs_level_remaining[ ] and coeff_sign_flag[ ] and sets ivlCurrRange to 256.

Renormalization is included in the specification of this bypass encoding process as given in Figure 9-13.

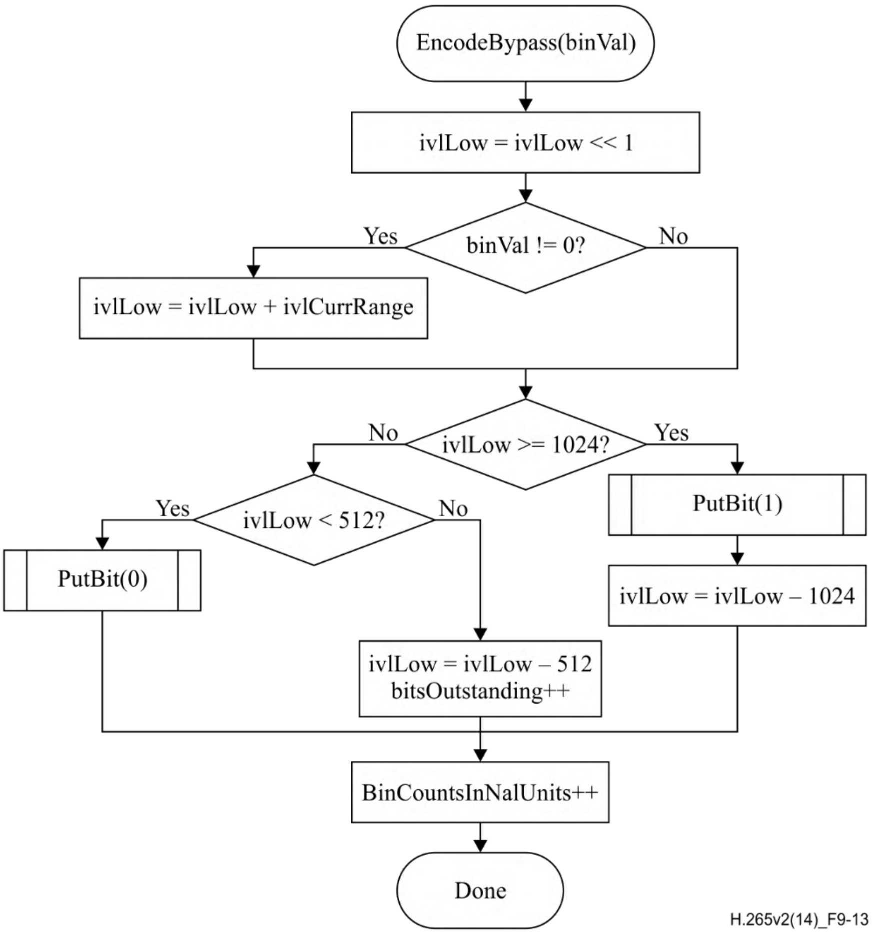


Figure 9-13 – Flowchart of encoding bypass

9.3.5.6 Encoding process for a binary decision before termination (informative)

This clause does not form an integral part of this Specification.

Inputs to this process are the variables binVal, ivlCurrRange, ivlLow, bitsOutstanding and BinCountsInNalUnits.

Outputs of this process are zero or more bits written to the RBSP and the updated variables ivlLow, ivlCurrRange, bitsOutstanding and BinCountsInNalUnits.

This encoding routine shown in Figure 9-14 applies to encoding of end_of_slice_segment_flag, end_of_subset_one_bit, and pcm_flag, all associated with ctxIdx equal to 0.

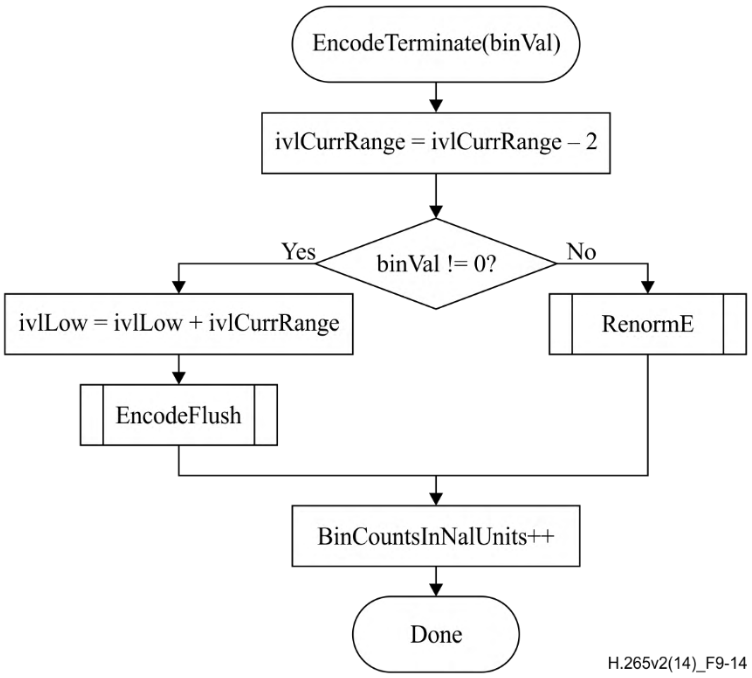


Figure 9-14 – Flowchart of encoding a decision before termination

When the value of binVal to encode is equal to 1, CABAC encoding is terminated and the flushing procedure shown in Figure 9-15 is applied. In this flushing procedure, the last bit written by WriteBits(B, N) is equal to 1. When encoding end_of_slice_segment_flag, this last bit is interpreted as rbsp_stop_one_bit. When encoding end_of_subset_one_bit, this last bit is interpreted as alignment_bit_equal_to_one.

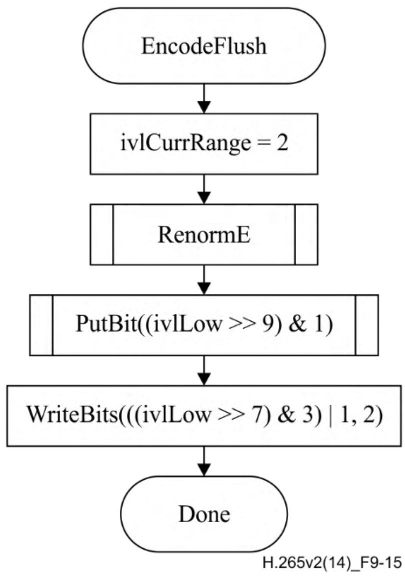


Figure 9-15 – Flowchart of flushing at termination

9.3.5.7 Byte stuffing process (informative)

This clause does not form an integral part of this Specification.

This process is invoked after encoding the last coding block of the last slice segment of a picture and after encapsulation.

Inputs to this process are the number of bytes NumBytesInVclNalUnits of all VCL NAL units of a picture, the number of minimum CUs PicSizeInMinCbsY in the picture and the number of binary symbols BinCountsInNalUnits resulting from encoding the contents of all VCL NAL units of the picture.

NOTE – The value of BinCountsInNalUnits is the result of encoding all slice segments of a coded picture. After initializing for the first slice segment of a coded picture as specified in clause 9.3.5.2, BinCountsInNalUnits is incremented as specified in clauses 9.3.5.3, 9.3.5.5 and 9.3.5.6.

Outputs of this process are zero or more bytes appended to the NAL unit.

Let the variable k be set equal to Ceil((Ceil(3 * (32 * BinCountsInNalUnits − RawMinCuBits * PicSizeInMinCbsY) ÷ 1 024) − NumBytesInVclNalUnits) ÷ 3). Depending on the value of k the following applies:

– If k is less than or equal to 0, no cabac_zero_word is appended to the NAL unit.

– Otherwise (k is greater than 0), the 3-byte sequence 0x000003 is appended k times to the NAL unit after encapsulation, where the first two bytes 0x0000 represent a cabac_zero_word and the third byte 0x03 represents an emulation_prevention_three_byte.

10 Sub-bitstream extraction process

Inputs to this process are a bitstream, a target highest TemporalId value tIdTarget and a target layer identifier list layerIdListTarget.

Output of this process is a sub-bitstream.

It is a requirement of bitstream conformance for the input bitstream that any output sub-bitstream that is the output of the process specified in this clause with the bitsteam, tIdTarget equal to any value in the range of 0 to 6, inclusive, and layerIdListTarget either equal to the layer identifier list associated with a layer set specified in the active VPS or consisting of all the nuh_layer_id values of the VCL NAL units present in the input bitstream as inputs, and that satisfies both of the following conditions shall be a conforming bitstream:

– The output sub-bitstream contains at least one VCL NAL unit with nuh_layer_id equal to each of the nuh_layer_id values in layerIdListTarget.

– The output sub-bitstream contains at least one VCL NAL unit with TemporalId equal to tIdTarget.

NOTE 1 – A bitstream conforming to a profile specified in Annex A contains one or more coded slice segment NAL units with nuh_layer_id equal to 0.

NOTE 2 – A conforming bitstream contains one or more coded slice segment NAL units with TemporalId equal to 0.

The output sub-bitstream is derived as follows:

– When one or more of the following two conditions are true, remove all SEI NAL units that have nuh_layer_id equal to 0 and that contain a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, or a non-scalable-nested decoding unit information SEI message:

  – layerIdListTarget does not include all the values of nuh_layer_id in all NAL units in the bitstream.

  – tIdTarget is less than the greatest TemporalId in all NAL units in the bitstream.

  NOTE 3 – A "smart" bitstream extractor may include appropriate non-scalable-nested buffering picture SEI messages, non-scalable-nested picture timing SEI messages and non-scalable-nested decoding unit information SEI messages in the extracted sub-bitstream, provided that the SEI messages applicable to the sub-bitstream were present as scalable-nested SEI messages in the original bitstream.

– Remove all NAL units with TemporalId greater than tIdTarget or nuh_layer_id not among the values included in layerIdListTarget.

Annex A

Profiles, tiers and levels

(This annex forms an integral part of this Recommendation | International Standard.)

A.1 Overview of profiles, tiers and levels

Profiles, tiers and levels specify restrictions on the bitstreams and hence limits on the capabilities needed to decode the bitstreams. Profiles, tiers and levels may also be used to indicate interoperability points between individual decoder implementations.

NOTE 1 – This Specification does not include individually selectable "options" at the decoder, as this would increase interoperability difficulties.

Each profile specifies a subset of algorithmic features and limits that shall be supported by all decoders conforming to that profile.

NOTE 2 – Encoders are not required to make use of any particular subset of features supported in a profile.

Each level of a tier specifies a set of limits on the values that may be taken by the syntax elements of this Specification. The same set of tier and level definitions is used with all profiles, but individual implementations may support a different tier and within a tier a different level for each supported profile. For any given profile, a level of a tier generally corresponds to a particular decoder processing load and memory capability.

The profiles that are specified in clause A.3 are also referred to as the profiles specified in Annex A.

A.2 Requirements on video decoder capability

Capabilities of video decoders conforming to this Specification are specified in terms of the ability to decode video streams conforming to the constraints of profiles, tiers and levels specified in this annex and other annexes. When expressing the capabilities of a decoder for a specified profile, the tier and level supported for that profile should also be expressed.

Specific values are specified in this annex and other annexes for the syntax elements general_profile_idc, general_tier_flag, general_level_idc, sub_layer_profile_idc[i], sub_layer_tier_flag[i] and sub_layer_level_idc[i]. All other values of general_profile_idc, general_level_idc, sub_layer_profile_idc[i] and sub_layer_level_idc[i] are reserved for future use by ITU-T | ISO/IEC.

NOTE – Decoders should not infer that a reserved value of general_profile_idc or sub_layer_profile_idc[i] between the values specified in this Specification indicates intermediate capabilities between the specified profiles, as there are no restrictions on the method to be chosen by ITU-T | ISO/IEC for the use of such future reserved values. However, decoders should infer that a reserved value of general_level_idc or sub_layer_level_idc[i] associated with a particular value of general_tier_flag or sub_layer_tier_flag[i], respectively, between the values specified in this Specification indicates intermediate capabilities between the specified levels of the tier.

A.3 Profiles

A.3.1 General

All constraints for PPSs that are specified are constraints for PPSs that are activated when the bitstream is decoded. All constraints for SPSs that are specified are constraints for SPSs that are activated when the bitstream is decoded.

The variable RawCtuBits is derived as follows:

$$\begin{aligned} \text{RawCtuBits} = {} & \text{CtbSizeY} * \text{CtbSizeY} * \text{BitDepth}_\text{Y} + \\ & 2 * (\text{CtbWidthC} * \text{CtbHeightC}) * \text{BitDepth}_\text{C} \end{aligned} \quad \text{(A-1)}$$

A.3.2 Main profile

Bitstreams conforming to the Main profile shall obey the following constraints:

– Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.

– Active SPSs for the base layer shall have chroma_format_idc equal to 1 only.

– Active SPSs for the base layer shall have bit_depth_luma_minus8 equal to 0 only.

– Active SPSs for the base layer shall have bit_depth_chroma_minus8 equal to 0 only.

– Active SPSs for the base layer shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag,

extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag, cabac_bypass_alignment_enabled_flag, sps_curr_pic_ref_enabled_flag, palette_mode_enabled_flag, motion_vector_resolution_control_idc, and intra_boundary_filtering_disabled_flag, when present, equal to 0 only.

– CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.

– Active PPSs for the base layer shall have log2_max_transform_skip_block_size_minus2, chroma_qp_offset_list_enabled_flag, and residual_adaptive_colour_transform_enabled_flag, when present, equal to 0 only.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the Main profile in clause A.4 shall be fulfilled.

Conformance of a bitstream to the Main profile is indicated by general_profile_idc being equal to 1 or general_profile_compatibility_flag[1] being equal to 1. Conformance of a sub-layer representation with TemporalId equal to i to the Main profile is indicated by sub_layer_profile_idc[i] being equal to 1 or sub_layer_profile_compatibility_flag[i][1] being equal to 1.

NOTE – When general_profile_compatibility_flag[1] is equal to 1, general_profile_compatibility_flag[2] should also be equal to 1. When sub_layer_profile_compatibility_flag[i][1] is equal to 1 for a value of i, sub_layer_profile_compatibility_flag[i][2] should also be equal to 1.

Decoders conforming to the Main profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

– The bitstream or sub-layer representation is indicated to conform to the Main profile or the Main Still Picture profile.

– The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.

– The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.3.3 Main 10 and Main 10 Still Picture profiles

Bitstreams conforming to the Main 10 or Main 10 Still Picture profile shall obey the following constraints:

– In bitstreams conforming to the Main 10 Still Picture profile, the bitstream shall contain only one picture with nuh_layer_id equal to 0.

– Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.

– Active SPSs for the base layer shall have chroma_format_idc equal to 1 only.

– Active SPSs for the base layer shall have bit_depth_luma_minus8 in the range of 0 to 2, inclusive.

– Active SPSs for the base layer shall have bit_depth_chroma_minus8 in the range of 0 to 2, inclusive.

– In bitstreams conforming to the Main 10 Still Picture profile, active SPSs for the base layer shall have sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] equal to 0 only.

– Active SPSs for the base layer shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag, extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag, cabac_bypass_alignment_enabled_flag, sps_curr_pic_ref_enabled_flag, palette_mode_enabled_flag, motion_vector_resolution_control_idc, and intra_boundary_filtering_disabled_flag, when present, equal to 0 only.

– CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.

– Active PPSs for the base layer shall have log2_max_transform_skip_block_size_minus2, chroma_qp_offset_list_enabled_flag, and residual_adaptive_colour_transform_enabled_flag, when present, equal to 0 only.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– In bitstreams conforming to the Main 10 profile that do not conform to the Main 10 Still Picture profile, general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the Main 10 or Main 10 Still Picture profile in clause A.4, as applicable, shall be fulfilled.

Conformance of a bitstream to the Main 10 profile is indicated by general_profile_idc being equal to 2 or general_profile_compatibility_flag[2] being equal to 1. Conformance of a sub-layer representation with TemporalId equal to i to the Main 10 profile is indicated by sub_layer_profile_idc[i] being equal to 2 or sub_layer_profile_compatibility_flag[i][2] being equal to 1.

Conformance of a bitstream to the Main 10 Still Picture profile is indicated by general_one_picture_only_constraint_flag being equal to 1 together with general_profile_idc being equal to 2 or general_profile_compatibility_flag[2] being equal to 1. Conformance of a sub-layer representation with TemporalId equal to i to the Main 10 Still Picture profile is indicated by sub_layer_one_picture_only_constraint_flag being equal to 1 together with sub_layer_profile_idc[i] being equal to 2 or sub_layer_profile_compatibility_flag[i][2] being equal to 1.

NOTE – When the conformance of a bitstream to the Main 10 Still Picture profile is indicated as specified above, and the indicated level is not level 8.5, the conditions for indication of the conformance of the bitstream to the Main 10 profile are also fulfilled.

Decoders conforming to the Main 10 profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

– The bitstream or sub-layer representation is indicated to conform to the Main 10 profile, the Main profile or the Main Still Picture profile.

– The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.

– The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

Decoders conforming to the Main 10 Still Picture profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

– The bitstream or sub-layer representation is indicated to conform to the Main 10 Still Picture profile or the Main Still Picture profile.

– The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.

– The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.3.4 Main Still Picture profile

Bitstreams conforming to the Main Still Picture profile shall obey the following constraints:

– The bitstream shall contain only one picture with nuh_layer_id equal to 0.

– Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.

- Active SPSs for the base layer shall have chroma_format_idc equal to 1 only.
- Active SPSs for the base layer shall have bit_depth_luma_minus8 equal to 0 only.
- Active SPSs for the base layer shall have bit_depth_chroma_minus8 equal to 0 only.
- Active SPSs for the base layer shall have sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] equal to 0 only.
- Active SPSs for the base layer shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag, extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag, cabac_bypass_alignment_enabled_flag, sps_curr_pic_ref_enabled_flag, palette_mode_enabled_flag, motion_vector_resolution_control_idc, and intra_boundary_filtering_disabled_flag, when present, equal to 0 only.
- CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- Active PPSs for the base layer shall have log2_max_transform_skip_block_size_minus2, chroma_qp_offset_list_enabled_flag, and residual_adaptive_colour_transform_enabled_flag, when present, equal to 0 only.
- When an active PPS for the base layer has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.
- When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.
- The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.
- The tier and level constraints specified for the Main Still Picture profile in clause A.4 shall be fulfilled.

Conformance of a bitstream to the Main Still Picture profile is indicated by general_profile_idc being equal to 3 or general_profile_compatibility_flag[3] being equal to 1.

NOTE – When general_profile_compatibility_flag[3] is equal to 1, general_profile_compatibility_flag[1] and general_profile_compatibility_flag[2] should also be equal to 1. When sub_layer_profile_compatibility_flag[i][3] is equal to 1 for a value of i, sub_layer_profile_compatibility_flag[i][1] and sub_layer_profile_compatibility_flag[i][2] should also be equal to 1.

Decoders conforming to the Main Still Picture profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams for which all of the following conditions apply:

- general_profile_idc is equal to 3 or general_profile_compatibility_flag[3] is equal to 1.
- general_level_idc is not equal to 255 and represents a level lower than or equal to the specified level.
- general_tier_flag represents a tier lower than or equal to the specified tier.

A.3.5 Format range extensions profiles

The following profiles, collectively referred to as the format range extensions profiles, are specified in this clause:

- The Monochrome, Monochrome 10, Monochrome 12 and Monochrome 16 profiles
- The Main 12 profile
- The Main 4:2:2 10 and Main 4:2:2 12 profiles
- The Main 4:4:4, Main 4:4:4 10 and Main 4:4:4 12 profiles
- The Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra and Main 4:4:4 16 Intra profiles
- The Main 4:4:4 Still Picture and Main 4:4:4 16 Still Picture profiles

Bitstreams conforming to the format range extensions profiles shall obey the following constraints:

- The constraints specified in Table A.1 shall apply, in which entries marked with "–" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element.

  NOTE – For some syntax elements with table entries marked with "–", a constraint may be imposed indirectly – e.g., by semantics constraints that are imposed elsewhere in this Specification when other specified constraints are fulfilled.

– Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.

– Active SPSs for the base layer shall have separate_colour_plane_flag, cabac_bypass_alignment_enabled_flag, sps_curr_pic_ref_enabled_flag, palette_mode_enabled_flag, motion_vector_resolution_control_idc, and intra_boundary_filtering_disabled_flag, when present, equal to 0 only.

– CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.

– Active PPSs for the base layer shall have residual_adaptive_colour_transform_enabled_flag, when present, equal to 0 only.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– In bitstreams conforming to the Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra or Main 4:4:4 16 Intra profiles, all pictures with nuh_layer_id equal to 0 shall be IRAP pictures and the output order indicated in the bitstream among these pictures shall be the same as the decoding order.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– In bitstreams conforming to the Main 4:4:4 Still Picture and Main 4:4:4 16 Still Picture profiles, the following constraints shall apply:

  – The bitstream shall contain only one picture with nuh_layer_id equal to 0.

  – Active SPSs for the base layer shall have sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] equal to 0 only.

– In bitstreams conforming to the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra or Main 4:4:4 16 Intra profiles, general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra or Main 4:4:4 16 Intra profiles in clause A.4, as applicable, shall be fulfilled.

Table A.1 – Allowed values for syntax elements in the format range extensions profiles

Profile for which constraint is specified	chroma_format_idc	bit_depth_luma_minus8 and bit_depth_chroma_minus8	transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag, intra_smoothing_disabled_flag, persistent_rice_adaptation_enabled_flag and log2_max_transform_skip_block_size_minus2	extended_precision_processing_flag	chroma_qp_offset_list_enabled_flag
Monochrome	0	0	0	0	0
Monochrome 10	0	0..2	0	0	0
Monochrome 12	0	0..4	0	0	0
Monochrome 16	0	–	–	–	0
Main 12	0 or 1	0..4	0	0	0
Main 4:2:2 10	0..2	0..2	0	0	–
Main 4:2:2 12	0..2	0..4	0	0	–
Main 4:4:4	–	0	–	0	–
Main 4:4:4 10	–	0..2	–	0	–
Main 4:4:4 12	–	0..4	–	0	–
Main Intra	0 or 1	0	0	0	0
Main 10 Intra	0 or 1	0..2	0	0	0
Main 12 Intra	0 or 1	0..4	0	0	0
Main 4:2:2 10 Intra	0..2	0..2	0	0	–
Main 4:2:2 12 Intra	0..2	0..4	0	0	–
Main 4:4:4 Intra	–	0	–	0	–
Main 4:4:4 10 Intra	–	0..2	–	0	–
Main 4:4:4 12 Intra	–	0..4	–	0	–
Main 4:4:4 16 Intra	–	–	–	–	–
Main 4:4:4 Still Picture	–	0	–	0	–
Main 4:4:4 16 Still Picture	–	–	–	–	–

Conformance of a bitstream to the format range extensions profiles is indicated by general_profile_idc being equal to 4 or general_profile_compatibility_flag[4] being equal to 1 with the additional indications specified in Table A.2. Conformance of a sub-layer representation with TemporalId equal to i to the format range extensions profiles is indicated by sub_layer_profile_idc[i] being equal to 4 or sub_layer_profile_compatibility_flag[i][4] being equal to 1 with the additional indications specified in Table A.2, with each of the syntax elements in Table A.2 being replaced by its i-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.2 with general_profile_idc equal to 4 or general_profile_compatibility_flag[4] equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the i-th corresponding sub-layer syntax elements of the syntax elements in Table A.2 with sub_layer_profile_idc[i] equal to 4 or sub_layer_profile_compatibility_flag[i][4] equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the format range extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.2 – Bitstream indications for conformance to format range extensions profiles

Profile for which the bitstream indicates conformance	general_max_12bit_constraint_flag	general_max_10bit_constraint_flag	general_max_8bit_constraint_flag	general_max_422chroma_constraint_flag	general_max_420chroma_constraint_flag	general_max_monochrome_constraint_flag	general_intra_constraint_flag	general_one_picture_only_constraint_flag	general_lower_bit_rate_constraint_flag
Monochrome	1	1	1	1	1	1	0	0	1
Monochrome 10	1	1	0	1	1	1	0	0	1
Monochrome 12	1	0	0	1	1	1	0	0	1
Monochrome 16	0	0	0	1	1	1	0	0	1
Main 12	1	0	0	1	1	0	0	0	1
Main 4:2:2 10	1	1	0	1	0	0	0	0	1
Main 4:2:2 12	1	0	0	1	0	0	0	0	1
Main 4:4:4	1	1	1	0	0	0	0	0	1
Main 4:4:4 10	1	1	0	0	0	0	0	0	1
Main 4:4:4 12	1	0	0	0	0	0	0	0	1
Main Intra	1	1	1	1	1	0	1	0	0 or 1
Main 10 Intra	1	1	0	1	1	0	1	0	0 or 1
Main 12 Intra	1	0	0	1	1	0	1	0	0 or 1
Main 4:2:2 10 Intra	1	1	0	1	0	0	1	0	0 or 1
Main 4:2:2 12 Intra	1	0	0	1	0	0	1	0	0 or 1
Main 4:4:4 Intra	1	1	1	0	0	0	1	0	0 or 1
Main 4:4:4 10 Intra	1	1	0	0	0	0	1	0	0 or 1
Main 4:4:4 12 Intra	1	0	0	0	0	0	1	0	0 or 1
Main 4:4:4 16 Intra	0	0	0	0	0	0	1	0	0 or 1
Main 4:4:4 Still Picture	1	1	1	0	0	0	1	1	0 or 1
Main 4:4:4 16 Still Picture	0	0	0	0	0	0	1	1	0 or 1

Decoders conforming to a format range extensions profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

– Any of the following conditions apply:
  – The decoder conforms to the Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4, Main 4:4:4 10 or Main 4:4:4 12 profile, and the bitstream or sub-layer representation is indicated to conform to the Main profile or the Main Still Picture profile.
  – The decoder conforms to the Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4 10 or Main 4:4:4 12 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 10 profile, the Main profile or the Main Still Picture profile.
  – The decoder conforms to the Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, or Main 4:4:4 12 Intra, Main 4:4:4 16 Intra, Main 4:4:4 Still Picture, or Main 4:4:4 16 Still Picture profile, and the bitstream or sub-layer representation is indicated to conform to the Main Still Picture profile.
  – general_profile_idc is equal to 4 or general_profile_compatibility_flag[4] is equal to 1 for the bitstream, and the value of each constraint flag listed in Table A.2 is greater than or equal to the value(s) specified in the row of Table A.2 for the format range extensions profile for which the decoder conformance is evaluated.
  – sub_layer_profile_idc[i] is equal to 4 or sub_layer_profile_compatibility_flag[i][4] is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.2 is greater than or equal to the value(s) specified in the row of Table A.2 for the format range extensions profile for which the decoder conformance is evaluated, with each of the syntax elements in Table A.2 being replaced by its i-th corresponding sub-layer syntax element.
– The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
– The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

For decoders conforming to the Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra, Main 4:4:4 16 Intra, Main 4:4:4 Still Picture, or Main 4:4:4 16 Still picture profile, the application of either or both of the in-loop filters of the in-loop filter process specified in clause 8.7 is optional.

A.3.6 High throughput profiles

The following profiles, collectively referred to as the high throughput profiles, are specified in this clause:

– The High Throughput 4:4:4, High Throughput 4:4:4 10 and High Throughput 4:4:4 14 profiles
– The High Throughput 4:4:4 16 Intra profile

NOTE 1 – For purposes of this terminology, the high-throughput screen content coding extensions profiles specified in clause A.3.8 are not included in the set of profiles that are collectively referred to as the high throughput profiles, although the names of some of the high-throughput screen content coding extensions profiles include the term "High Throughput".

Bitstreams conforming to the high throughput profiles shall obey the following constraints:

– Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.
– Active SPSs for the base layer shall have separate_colour_plane_flag, sps_curr_pic_ref_enabled_flag, palette_mode_enabled_flag, motion_vector_resolution_control_idc, and intra_boundary_filtering_disabled_flag, when present, equal to 0 only.
– In bitstreams conforming to the High Throughput 4:4:4 profile, active SPSs for the base layer shall have bit_depth_luma_minus8 equal to 0, bit_depth_chroma_minus8 equal to 0, extended_precision_processing_flag equal to 0, and cabac_bypass_alignment_enabled_flag equal to 0 only.
– In bitstreams conforming to the High Throughput 4:4:4 10 profile, active SPSs for the base layer shall have bit_depth_luma_minus8 less than or equal to 2, bit_depth_chroma_minus8 less than or equal to 2, extended_precision_processing_flag equal to 0, and cabac_bypass_alignment_enabled_flag equal to 0 only.
– In bitstreams conforming to the High Throughput 4:4:4 14 profile, active SPSs for the base layer shall have bit_depth_luma_minus8 less than or equal to 6 and bit_depth_chroma_minus8 less than or equal to 6.

– In bitstreams conforming to the High Throughput 4:4:4 16 Intra profile, active SPSs for the base layer shall have cabac_bypass_alignment_enabled_flag equal to 1 only.

– CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.

– Active PPSs for the base layer shall have residual_adaptive_colour_transform_enabled_flag, when present, equal to 0 only.

– In bitstreams conforming to the High Throughput 4:4:4, High Throughput 4:4:4 10, or High Throughput 4:4:4 14 profiles, active PPSs for the base layer shall have entropy_coding_sync_enabled_flag equal to 1 only.

NOTE 2 – Unlike for some other profiles specified in this annex, an active PPS for the base layer for the high throughput profiles may have tiles_enabled_flag equal to 1 with entropy_coding_sync_enabled_flag equal to 1.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– In bitstreams conforming to the High Throughput 4:4:4 16 Intra profile, all pictures with nuh_layer_id equal to 0 shall be IRAP pictures and the output order indicated in the bitstream among these pictures shall be the same as the decoding order.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the High Throughput 4:4:4, High Throughput 4:4:4 10, High Throughput 4:4:4 14 or High Throughput 4:4:4 16 Intra profile in clause A.4, as applicable, shall be fulfilled.

Conformance of a bitstream to the high throughput profiles is indicated by general_profile_idc being equal to 5 or general_profile_compatibility_flag[5] being equal to 1 with the additional indications specified in Table A.3. Conformance of a sub-layer representation with TemporalId equal to i to the high throughput profiles is indicated by sub_layer_profile_idc[i] being equal to 5 or sub_layer_profile_compatibility_flag[i][5] being equal to 1 with the additional indications specified in Table A.3, with each of the syntax elements in Table A.3 being replaced by its i-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.3 with general_profile_idc equal to 5 or general_profile_compatibility_flag[5] equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the i-th corresponding sub-layer syntax elements of the syntax elements in Table A.3 with sub_layer_profile_idc[i] equal to 5 or sub_layer_profile_compatibility_flag[i][5] equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the format range extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.3 – Bitstream indications for conformance to high throughput profiles

Profile for which the bitstream indicates conformance	general_max_14bit_constraint_flag	general_max_12bit_constraint_flag	general_max_10bit_constraint_flag	general_max_8bit_constraint_flag	general_max_422chroma_constraint_flag	general_max_420chroma_constraint_flag	general_max_monochrome_constraint_flag	general_intra_constraint_flag	general_one_picture_only_constraint_flag	general_lower_bit_rate_constraint_flag
High Throughput 4:4:4	1	1	1	1	0	0	0	0	0	1
High Throughput 4:4:4 10	1	1	1	0	0	0	0	0	0	1
High Throughput 4:4:4 14	1	0	0	0	0	0	0	0	0	1
High Throughput 4:4:4 16 Intra	0	0	0	0	0	0	0	1	0	0 or 1

Decoders conforming to a high throughput profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams or sub-layer representations for which all of the following conditions apply:

– Any of the following conditions apply:

  – general_profile_idc is equal to 5 or general_profile_compatibility_flag[5] is equal to 1 for the bitstream and the value of each constraint flag listed in Table A.3 is greater than or equal to the value(s) specified in the row of Table A.3 for the high throughput profile for which the decoder conformance is evaluated.

  – sub_layer_profile_idc[i] is equal to 5 or sub_layer_profile_compatibility_flag[i][5] is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.3 is greater than or equal to the value(s) specified in the row of Table A.3 for the high throughput profile for which the decoder conformance is evaluated, with each of the syntax elements in Table A.3 being replaced by its i-th corresponding sub-layer syntax element.

– The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.

– The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

For decoders conforming to the High Throughput 4:4:4 16 Intra profile, the application of either or both of the in-loop filters of the in-loop filter process specified in clause 8.7 is optional.

A.3.7 Screen content coding extensions profiles

The following profiles, collectively referred to as the screen content coding extensions profiles, are specified in this clause:

– The Screen-Extended Main and Screen-Extended Main 10 profiles

– The Screen-Extended Main 4:4:4 and Screen-Extended Main 4:4:4 10 profiles

NOTE – For purposes of this terminology, the high throughput screen content coding extensions profiles specified in clause A.3.8 are not included in the set of profiles that are collectively referred to as the screen content coding extensions profiles, although the names of some of the high throughput screen content coding extensions profiles include the term "Screen-Extended".

Bitstreams conforming to the screen content coding extensions profiles shall obey the following constraints:

– The constraints specified in Table A.4 shall apply, in which entries marked with "–" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element.

– Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.

– Active SPSs for the base layer shall have separate_colour_plane_flag, when present, equal to 0 only.

– CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.

– When an active SPS for the base layer has palette_mode_enabled_flag equal to 1, palette_max_size shall be less than or equal to 64 and PaletteMaxPredictorSize shall be less than or equal to 128.

– In bitstreams conforming to the Screen-Extended Main, Screen-Extended Main 10, Screen-Extended Main 4:4:4, or Screen-Extended Main 4:4:4 10 profiles, active SPSs for the base layer shall have extended_precision_processing_flag, and cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.

– In bitstreams conforming to the Screen-Extended Main or Screen-Extended Main 10 profiles, when an active PPS for the base layer has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the Screen-Extended Main, Screen-Extended Main 10, Screen-Extended Main 4:4:4 or Screen-Extended Main 4:4:4 10 profiles in clause A.4, as applicable, shall be fulfilled.

Table A.4 – Allowed values for syntax elements in the screen content coding extensions profiles

Profile for which constraint is specified	**chroma_format_idc**	**bit_depth_luma_minus8** and **bit_depth_chroma_minus8**
Screen-Extended Main	1	0
Screen-Extended Main 10	1	0..2
Screen-Extended Main 4:4:4	0, 1, or 3	0
Screen-Extended Main 4:4:4 10	0, 1, or 3	0..2

Conformance of a bitstream to the screen content coding extensions profiles is indicated by general_profile_idc being equal to 9 or general_profile_compatibility_flag[9] being equal to 1 with the additional indications specified in Table A.5. Conformance of a sub-layer representation with TemporalId equal to i to the screen content coding extensions profiles is indicated by sub_layer_profile_idc[i] being equal to 9 or sub_layer_profile_compatibility_flag[i][9] being equal to 1 with the additional indications specified in Table A.5, with each of the syntax elements in Table A.5 being replaced by its i-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.5 with general_profile_idc equal to 9 or general_profile_compatibility_flag[9] equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the i-th corresponding sub-layer syntax elements of the syntax elements in Table A.5 with sub_layer_profile_idc[i] equal to 9 or sub_layer_profile_compatibility_flag[i][9] equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the screen content coding extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.5 – Bitstream indications for conformance to screen content coding extensions profiles

Profile for which the bitstream indicates conformance	general_max_14bit_constraint_flag	general_max_12bit_constraint_flag	general_max_10bit_constraint_flag	general_max_8bit_constraint_flag	general_max_422chroma_constraint_flag	general_max_420chroma_constraint_flag	general_max_monochrome_constraint_flag	general_intra_constraint_flag	general_one_picture_only_constraint_flag	general_lower_bit_rate_constraint_flag
Screen-Extended Main	1	1	1	1	1	1	0	0	0	1
Screen-Extended Main 10	1	1	1	0	1	1	0	0	0	1
Screen-Extended Main 4:4:4	1	1	1	1	0	0	0	0	0	1
Screen-Extended Main 4:4:4 10	1	1	1	0	0	0	0	0	0	1

Decoders conforming to a screen content coding extensions profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

– Any of the following conditions apply:
  – The bitstream or sub-layer representation is indicated to conform to the Main, Main Still Picture, or Monochrome profile.
  – The decoder conforms to the Screen-Extended Main 10 or Screen-Extended Main 4:4:4 10 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 10 profile.
  – The decoder conforms to the Screen-Extended Main 4:4:4 or Screen-Extended Main 4:4:4 10 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 4:4:4 profile.
  – The decoder conforms to the Screen-Extended Main 4:4:4 10 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 4:4:4 10 profile.
  – general_profile_idc is equal to 4 or general_profile_compatibility_flag[4] is equal to 1 or general_profile_idc is equal to 9 or general_profile_compatibility_flag[9] is equal to 1 for the bitstream, and the value of each constraint flag listed in Table A.5 is greater than or equal to the value(s) specified in the row of Table A.5 for the screen content coding extensions profile for which the decoder conformance is evaluated, and general_max_422chroma_constraint_flag is equal to general_max_420chroma_constraint_flag.
  – sub_layer_profile_idc[i] is equal to 4 or sub_layer_profile_compatibility_flag[i][4] is equal to 1 or sub_layer_profile_idc[i] is equal to 9 or sub_layer_profile_compatibility_flag[i][9] is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.5 is greater than or equal to the value(s) specified in the row of Table A.5 for the screen content coding extensions profile for which the decoder conformance is evaluated, and general_max_422chroma_constraint_flag is equal to general_max_420chroma_constraint_flag, with each of the syntax elements in Table A.5 being replaced by its i-th corresponding sub-layer syntax element.

– The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.

– The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.3.8 High throughput screen content coding extensions profiles

The following profiles, collectively referred to as the high throughput screen content coding extensions profiles, are specified in this clause:

– The Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, and Screen-Extended High Throughput 14 profiles

Bitstreams conforming to the screen content coding extensions profiles shall obey the following constraints:

– The constraints specified in Table A.6 shall apply, in which entries marked with "–" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element.

– Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.

– Active SPSs for the base layer shall have separate_colour_plane_flag, when present, equal to 0 only.

– CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.

– When an active SPS for the base layer has palette_mode_enabled_flag equal to 1, palette_max_size shall be less than or equal to 64 and PaletteMaxPredictorSize shall be less than or equal to 128.

– Active SPSs for the base layer shall have extended_precision_processing_flag, and cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.

– Active PPSs for the base layer shall have entropy_coding_sync_enabled_flag equal to 1 only.

NOTE – Unlike for some other profiles specified in this annex, an active PPS for the base layer for Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, or Screen-Extended High Throughput 4:4:4 14 profiles may have tiles_enabled_flag equal to 1 with entropy_coding_sync_enabled_flag equal to 1.

– When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, and Screen-Extended High Throughput 14 profiles in clause A.4, as applicable, shall be fulfilled.

Table A.6 – Allowed values for syntax elements in the high throughput screen content coding extensions profiles

Profile for which constraint is specified	**chroma_format_idc**	**bit_depth_luma_minus8** and **bit_depth_chroma_minus8**
Screen-Extended High Throughput 4:4:4	–	0
Screen-Extended High Throughput 4:4:4 10	–	0..2
Screen-Extended High Throughput 4:4:4 14	–	0..6

Conformance of a bitstream to the high throughput screen content coding extensions profiles is indicated by general_profile_idc being equal to 11 or general_profile_compatibility_flag[11] being equal to 1 with the additional indications specified in Table A.7. Conformance of a sub-layer representation with TemporalId equal to i to the screen content coding extensions profiles is indicated by sub_layer_profile_idc[i] being equal to 11 or sub_layer_profile_compatibility_flag[i][11] being equal to 1 with the additional indications specified in Table A.7, with each of the syntax elements in Table A.7 being replaced by its i-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.7 with general_profile_idc equal to 11 or general_profile_compatibility_flag[11] equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the i-th corresponding sub-layer syntax elements of the syntax elements in Table A.7 with sub_layer_profile_idc[i] equal to 11 or sub_layer_profile_compatibility_flag[i][11] equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the screen content coding extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.7 – Bitstream indications for conformance to high throughput screen content coding extensions profiles

Profile for which the bitstream indicates conformance	general_max_14bit_constraint_flag	general_max_12bit_constraint_flag	general_max_10bit_constraint_flag	general_max_8bit_constraint_flag	general_max_422chroma_constraint_flag	general_max_420chroma_constraint_flag	general_max_monochrome_constraint_flag	general_intra_constraint_flag	general_one_picture_only_constraint_flag	general_lower_bit_rate_constraint_flag
Screen-Extended High Throughput 4:4:4	1	1	1	1	0	0	0	0	0	1
Screen-Extended High Throughput 4:4:4 10	1	1	1	0	0	0	0	0	0	1
Screen-Extended High Throughput 4:4:4 14	1	0	0	0	0	0	0	0	0	1

Decoders conforming to a high throughput screen content coding extensions profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

– Any of the following conditions apply:
  – The bitstream or sub-layer representation is indicated to conform to the Main, Main Still Picture, or Monochrome profile.
  – The bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 profile.
  – The decoder conforms to the Screen-Extended High Throughput 4:4:4 10 or Screen-Extended High Throughput 4:4:4 14 profile, and the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 10 profile.
  – The decoder conforms to the Screen-Extended High Throughput 4:4:4 14 profile, and the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 14 profile.
  – general_profile_idc is equal to 4 or general_profile_compatibility_flag[4] is equal to 1 or general_profile_idc is equal to 11 or general_profile_compatibility_flag[11] is equal to 1 for the bitstream, and the value of each constraint flag listed in Table A.7 is greater than or equal to the value(s) specified in the row of Table A.7 for the screen content coding extensions profile for which the decoder conformance is evaluated, and general_max_422chroma_constraint_flag is equal to general_max_420chroma_constraint_flag.
  – sub_layer_profile_idc[i] is equal to 4 or sub_layer_profile_compatibility_flag[i][4] is equal to 1 or sub_layer_profile_idc[i] is equal to 11 or sub_layer_profile_compatibility_flag[i][11] is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.7 is greater than or equal to the value(s) specified in the row of Table A.7 for the screen content coding extensions profile for which the decoder conformance is evaluated, and general_max_422chroma_constraint_flag is equal to general_max_420chroma_constraint_flag, with each of the syntax elements in Table A.7 being replaced by its i-th corresponding sub-layer syntax element, respectively.
– The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
– The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.4 Tiers and levels

A.4.1 General tier and level limits

For purposes of comparison of tier capabilities, the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 0 is considered to be a lower tier than the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 1.

For purposes of comparison of level capabilities, a particular level of a specific tier is considered to be a lower level than some other level of the same tier when the value of the general_level_idc or sub_layer_level_idc[i] of the particular level is less than that of the other level.

The following is specified for expressing the constraints in this annex:

– Let access unit n be the n-th access unit in decoding order, with the first access unit being access unit 0 (i.e., the 0-th access unit).

– Let picture n be the coded picture or the corresponding decoded picture of access unit n.

When the specified level is not level 8.5, bitstreams conforming to a profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in Annex C:

a) PicSizeInSamplesY shall be less than or equal to MaxLumaPs, where MaxLumaPs is specified in Table A.8.

b) The value of pic_width_in_luma_samples shall be less than or equal to Sqrt(MaxLumaPs * 8).

c) The value of pic_height_in_luma_samples shall be less than or equal to Sqrt(MaxLumaPs * 8).

d) For level 5 and higher levels, the value of CtbSizeY shall be equal to 32 or 64.

e) The value of NumPicTotalCurr shall be less than or equal to 8.

f) The value of num_tile_columns_minus1 shall be less than MaxTileCols and num_tile_rows_minus1 shall be less than MaxTileRows, where MaxTileCols and MaxTileRows are specified in Table A.8.

g) For the VCL HRD parameters, CpbSize[i] shall be less than or equal to CpbVclFactor * MaxCPB for at least one value of i in the range of 0 to cpb_cnt_minus1[HighestTid], inclusive, where CpbSize[i] is specified in clause E.3.3 based on parameters selected as specified in clause C.1, CpbVclFactor is specified in Table A.10, and MaxCPB is specified in Table A.8 in units of CpbVclFactor bits.

h) For the NAL HRD parameters, CpbSize[i] shall be less than or equal to CpbNalFactor * MaxCPB for at least one value of i in the range of 0 to cpb_cnt_minus1[HighestTid], inclusive, where CpbSize[i] is specified in clause E.3.3 based on parameters selected as specified in clause C.1, CpbNalFactor is specified in Table A.10, and MaxCPB is specified in Table A.8 in units of CpbNalFactor bits.

Table A.8 specifies the limits for each level of each tier for levels other than level 8.5.

A tier and level to which a bitstream conforms are indicated by the syntax elements general_tier_flag and general_level_idc, and a tier and level to which a sub-layer representation conforms are indicated by the syntax elements sub_layer_tier_flag[i] and sub_layer_level_idc[i], as follows:

– If the specified level is not level 8.5, general_tier_flag or sub_layer_tier_flag[i] equal to 0 indicates conformance to the Main tier, general_tier_flag or sub_layer_tier_flag[i] equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8 and general_tier_flag and sub_layer_tier_flag[i] shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A.8 marked with "-"). Otherwise (the specified level is level 8.5), it is a requirement of bitstream conformance that general_tier_flag and sub_layer_tier_flag[i] shall be equal to 1 and the value 0 for general_tier_flag and sub_layer_tier_flag[i] is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of general_tier_flag and sub_layer_tier_flag[i].

– general_level_idc and sub_layer_level_idc[i] shall be set equal to a value of 30 times the level number specified in Table A.8.

Table A.8 – General tier and level limits

Level	Max luma picture size MaxLumaPs (samples)	Max CPB size MaxCPB (CpbVclFactor or CpbNalFactor bits)		Max slice segments per picture MaxSliceSegmentsPerPicture	Max # of tile rows MaxTileRows	Max # of tile columns MaxTileCols
		Main tier	High tier			
1	36 864	350	-	16	1	1
2	122 880	1 500	-	16	1	1
2.1	245 760	3 000	-	20	1	1
3	552 960	6 000	-	30	2	2
3.1	983 040	10 000	-	40	3	3
4	2 228 224	12 000	30 000	75	5	5
4.1	2 228 224	20 000	50 000	75	5	5
5	8 912 896	25 000	100 000	200	11	10
5.1	8 912 896	40 000	160 000	200	11	10
5.2	8 912 896	60 000	240 000	200	11	10
6	35 651 584	60 000	240 000	600	22	20
6.1	35 651 584	120 000	480 000	600	22	20
6.2	35 651 584	240 000	800 000	600	22	20

A.4.2 Profile-specific level limits for the video profiles

NOTE – The term "video profiles", as used in this clause, refers to those profiles that are not still picture profiles. The still picture profiles include the Main Still Picture, Main 10 Still Picture, Main 4:4:4 Still Picture, and Main 4:4:4 16 Still Picture profiles.

The following is specified for expressing the constraints in this annex:

– Let the variable fR be set equal to 1 ÷ 300.

The variable HbrFactor is defined as follows:

– If the bitstream or sub-layer representation is indicated to conform to the Main profile or the Main 10 profile, HbrFactor is set equal to 1.

– Otherwise, if the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4, High Throughput 4:4:4 10, High Throughput 4:4:4 14, Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, or Screen-Extended High Throughput 4:4:4 14 profile, HbrFactor is set equal to 6.

– Otherwise, if the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 16 Intra profile, HbrFactor is set equal to 24 − (12 * general_lower_bit_rate_constraint_flag) or 24 − (12 * sub_layer_lower_bit_rate_constraint_flag[i]).

– Otherwise, HbrFactor is set equal to 2 − general_lower_bit_rate_constraint_flag or 2 − sub_layer_lower_bit_rate_constraint_flag[i].

The variable BrVclFactor, which represents the VCL bit rate scale factor, is set equal to CpbVclFactor * HbrFactor.

The variable BrNalFactor, which represents the NAL bit rate scale factor, is set equal to CpbNalFactor * HbrFactor.

The variable MinCr is set equal to MinCrBase * MinCrScaleFactor ÷ HbrFactor.

When the specified level is not level 8.5, the value of sps_max_dec_pic_buffering_minus1[HighestTid] + 1 shall be less than or equal to MaxDpbSize, which is derived as follows:

```
if( PicSizeInSamplesY <= ( MaxLumaPs >> 2 ) )
    MaxDpbSize = Min( 4 * maxDpbPicBuf, 16 )
```

else if(PicSizeInSamplesY <= (MaxLumaPs >> 1))
    MaxDpbSize = Min(2 * maxDpbPicBuf, 16) (A-2)
else if(PicSizeInSamplesY <= ((3 * MaxLumaPs) >> 2))
    MaxDpbSize = Min((4 * maxDpbPicBuf) / 3, 16)
else
    MaxDpbSize = maxDpbPicBuf

where MaxLumaPs is specified in Table A.8, and maxDpbPicBuf is equal to 6 for all profiles where the value of sps_curr_pic_ref_enabled_flag is required to be equal to 0 and 7 for all profiles where the value of sps_curr_pic_ref_enabled_flag is not required to be equal to 0.

Bitstreams and sub-layer representations conforming to the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main, Main 10, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra, Main 4:4:4 16 Intra High Throughput 4:4:4, High Throughput 4:4:4 10, High Throughput 4:4:4 14, High Throughput 4:4:4 16 Intra , Screen-Extended Main, Screen-Extended Main 10, Screen-Extended Main 4:4:4, Screen-Extended Main 4:4:4 10, Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, or Screen-Extended High Throughput 4:4:4 14 profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in Annex C:

a) The nominal removal time of access unit n (with n greater than 0) from the CPB, as specified in clause C.2.3, shall satisfy the constraint that AuNominalRemovalTime[n] − AuCpbRemovalTime[n − 1] is greater than or equal to Max(PicSizeInSamplesY ÷ MaxLumaSr, fR) for the value of PicSizeInSamplesY of picture n − 1, where MaxLumaSr is the value specified in Table A.9 that applies to picture n − 1.

b) The difference between consecutive output times of pictures from the DPB, as specified in clause C.3.3, shall satisfy the constraint that DpbOutputInterval[n] is greater than or equal to Max(PicSizeInSamplesY ÷ MaxLumaSr, fR) for the value of PicSizeInSamplesY of picture n, where MaxLumaSr is the value specified in Table A.9 for picture n, provided that picture n is a picture that is output and is not the last picture of the bitstream that is output.

c) The removal time of access unit 0 shall satisfy the constraint that the number of slice segments in picture 0 is less than or equal to Min(Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0]) + MaxSliceSegmentsPerPicture * PicSizeInSamplesY / MaxLumaPs), MaxSliceSegmentsPerPicture), for the value of PicSizeInSamplesY of picture 0, where MaxSliceSegmentsPerPicture, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9, respectively, that apply to picture 0.

d) The difference between consecutive CPB removal times of access units n and n − 1 (with n greater than 0) shall satisfy the constraint that the number of slice segments in picture n is less than or equal to Min((Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1])), MaxSliceSegmentsPerPicture), where MaxSliceSegmentsPerPicture, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9 that apply to picture n.

e) For the VCL HRD parameters, BitRate[i] shall be less than or equal to BrVclFactor * MaxBR for at least one value of i in the range of 0 to cpb_cnt_minus1[HighestTid], inclusive, where BitRate[i] is specified in clause E.3.3 based on parameters selected as specified in clause C.1 and MaxBR is specified in Table A.9 in units of BrVclFactor bits/s.

f) For the NAL HRD parameters, BitRate[i] shall be less than or equal to BrNalFactor * MaxBR for at least one value of i in the range of 0 to cpb_cnt_minus1[HighestTid], inclusive, where BitRate[i] is specified in clause E.3.3 based on parameters selected as specified in clause C.1 and MaxBR is specified in Table A.9 in units of BrNalFactor bits/s.

g) The sum of the NumBytesInNalUnit variables for access unit 0 shall be less than or equal to FormatCapabilityFactor * (Max(PicSizeInSamplesY, fR * MaxLumaSr) + MaxLumaSr * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0])) ÷ MinCr for the value of PicSizeInSamplesY of picture 0, where MaxLumaSr and FormatCapabilityFactor are the values specified in Table A.9 and Table A.10, respectively, that apply to picture 0.

h) The sum of the NumBytesInNalUnit variables for access unit n (with n greater than 0) shall be less than or equal to FormatCapabilityFactor * MaxLumaSr * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1]) ÷ MinCr, where MaxLumaSr and FormatCapabilityFactor are the values specified in Table A.9 and Table A.10, respectively, that apply to picture n.

i) The removal time of access unit 0 shall satisfy the constraint that the number of tiles in picture 0 is less than or equal to Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[0] −

AuNominalRemovalTime[0]) + MaxTileCols * MaxTileRows * PicSizeInSamplesY / MaxLumaPs), MaxTileCols * MaxTileRows), for the value of PicSizeInSamplesY of picture 0, where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to picture 0.

j) The difference between consecutive CPB removal times of access units n and n − 1 (with n greater than 0) shall satisfy the constraint that the number of tiles in picture n is less than or equal to Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1])), MaxTileCols * MaxTileRows), where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to picture n.

k)

Table A.9 – Tier and level limits for the video profiles

Level	Max luma sample rate MaxLumaSr (samples/sec)	Max bit rate MaxBR (BrVclFactor or BrNalFactor bits/s)		Min compression ratio MinCrBase	
		Main tier	High tier	Main tier	High tier
1	552 960	128	-	2	2
2	3 686 400	1 500	-	2	2
2.1	7 372 800	3 000	-	2	2
3	16 588 800	6 000	-	2	2
3.1	33 177 600	10 000	-	2	2
4	66 846 720	12 000	30 000	4	4
4.1	133 693 440	20 000	50 000	4	4
5	267 386 880	25 000	100 000	6	4
5.1	534 773 760	40 000	160 000	8	4
5.2	1 069 547 520	60 000	240 000	8	4
6	1 069 547 520	60 000	240 000	8	4
6.1	2 139 095 040	120 000	480 000	8	4
6.2	4 278 190 080	240 000	800 000	6	4

Table A.10 – Specification of CpbVclFactor, CpbNalFactor, FormatCapabilityFactor and MinCrScaleFactor

Profile	CpbVclFactor	CpbNalFactor	FormatCapabilityFactor	MinCrScaleFactor
Monochrome	667	733	1.000	1.0
Monochrome 10	833	917	1.250	1.0
Monochrome 12	1 000	1 100	1.500	1.0
Monochrome 16	1 333	1 467	2.000	1.0
Main	1 000	1 100	1.500	1.0
Screen-Extended Main	1 000	1 100	1.500	1.0
Main 10	1 000	1 100	1.875	1.0
Screen-Extended Main 10	1 000	1 100	1.875	1.0
Main 12	1 500	1 650	2.250	1.0
Main Still Picture	1 000	1 100	1.500	1.0
Main 10 Still Picture	1 000	1 100	1.875	1.0
Main 4:2:2 10	1 667	1 833	2.500	0.5
Main 4:2:2 12	2 000	2 200	3.000	0.5
Main 4:4:4	2 000	2 200	3.000	0.5
High Throughput 4:4:4	2 000	2 200	3.000	0.5
Screen-Extended Main 4:4:4	2 000	2 200	3.000	0.5
Screen-Extended High Throughput 4:4:4	2 000	2 200	3.000	0.5
Main 4:4:4 10	2 500	2 750	3.750	0.5
High Throughput 4:4:4 10	2 500	2 750	3.750	0.5
Screen-Extended Main 4:4:4 10	2 500	2 750	3.750	0.5
Screen-Extended High Throughput 4:4:4 10	2 500	2 750	3.750	0.5
Main 4:4:4 12	3 000	3 300	4.500	0.5
High Throughput 4:4:4 14	3 500	3 850	5.250	0.5
Screen-Extended High Throughput 4:4:4 14	3 500	3 850	5.250	0.5
Main Intra	1 000	1 100	1.500	1.0
Main 10 Intra	1 000	1 100	1.875	1.0
Main 12 Intra	1 500	1 650	2.250	1.0
Main 4:2:2 10 Intra	1 667	1 833	2.500	0.5
Main 4:2:2 12 Intra	2 000	2 200	3.000	0.5
Main 4:4:4 Intra	2 000	2 200	3.000	0.5
Main 4:4:4 10 Intra	2 500	2 750	3.750	0.5
Main 4:4:4 12 Intra	3 000	3 300	4.500	0.5
Main 4:4:4 16 Intra	4 000	4 400	6.000	0.5
Main 4:4:4 Still Picture	2 000	2 200	3.000	0.5
Main 4:4:4 16 Still Picture	4 000	4 400	6.000	0.5
High Throughput 4:4:4 16 Intra	4 000	4 400	6.000	0.5

Informative clause A.4.3 shows the effect of these limits on picture rates for several example picture formats.

A.4.3 Effect of level limits on picture rate for the video profiles (informative)

This clause does not form an integral part of this Specification.

Informative Tables A.11 and A.12 provide examples of maximum picture rates for the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main, Main 10, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra, Main 4:4:4 16 Intra and High Throughput 4:4:4 16 Intra profiles for various picture formats when MinCbSizeY is equal to 64.

Table A.11 – Maximum picture rates (pictures per second) at level 1 to 4.1 for some example picture sizes when MinCbSizeY is equal to 64

Level:				**1**	**2**	**2.1**	**3**	**3.1**	**4**	**4.1**
Max luma picture size (samples):				36 864	122 880	245 760	552 960	983 040	2 228 224	2 228 224
Max luma sample rate (samples/sec)				552 960	3 686 400	7 372 800	16 588 800	33 177 600	66 846 720	133 693 440
Format nickname	**Luma width**	**Luma height**	**Luma picture size**							
SQCIF	**128**	**96**	16 384	33.7	225.0	300.0	300.0	300.0	300.0	300.0
QCIF	**176**	**144**	36 864	15.0	100.0	200.0	300.0	300.0	300.0	300.0
QVGA	**320**	**240**	81 920	-	45.0	90.0	202.5	300.0	300.0	300.0
525 SIF	**352**	**240**	98 304	-	37.5	75.0	168.7	300.0	300.0	300.0
CIF	**352**	**288**	122 880	-	30.0	60.0	135.0	270.0	300.0	300.0
525 HHR	**352**	**480**	196 608	-	-	37.5	84.3	168.7	300.0	300.0
625 HHR	**352**	**576**	221 184	-	-	33.3	75.0	150.0	300.0	300.0
Q720p	**640**	**360**	245 760	-	-	30.0	67.5	135.0	272.0	300.0
VGA	**640**	**480**	327 680	-	-	-	50.6	101.2	204.0	300.0
525 4SIF	**704**	**480**	360 448	-	-	-	46.0	92.0	185.4	300.0
525 SD	**720**	**480**	393 216	-	-	-	42.1	84.3	170.0	300.0
4CIF	**704**	**576**	405 504	-	-	-	40.9	81.8	164.8	300.0
625 SD	**720**	**576**	442 368	-	-	-	37.5	75.0	151.1	300.0
480p (16:9)	**864**	**480**	458 752	-	-	-	36.1	72.3	145.7	291.4
SVGA	**800**	**600**	532 480	-	-	-	31.1	62.3	125.5	251.0
QHD	**960**	**540**	552 960	-	-	-	30.0	60.0	120.8	241.7
XGA	**1 024**	**768**	786 432	-	-	-	-	42.1	85.0	170.0
720p HD	**1 280**	**720**	983 040	-	-	-	-	33.7	68.0	136.0
4VGA	**1 280**	**960**	1 228 800	-	-	-	-	-	54.4	108.8
SXGA	**1 280**	**1 024**	1 310 720	-	-	-	-	-	51.0	102.0
525 16SIF	**1 408**	**960**	1 351 680	-	-	-	-	-	49.4	98.9
16CIF	**1 408**	**1 152**	1 622 016	-	-	-	-	-	41.2	82.4
4SVGA	**1 600**	**1 200**	1 945 600	-	-	-	-	-	34.3	68.7
1080 HD	**1 920**	**1 080**	2 088 960	-	-	-	-	-	32.0	64.0
2Kx1K	**2 048**	**1 024**	2 097 152	-	-	-	-	-	31.8	63.7
2Kx1080	**2 048**	**1 080**	2 228 224	-	-	-	-	-	30.0	60.0
4XGA	**2 048**	**1 536**	3 145 728	-	-	-	-	-	-	-
16VGA	**2 560**	**1 920**	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	**3 616**	**1 536**	5 603 328	-	-	-	-	-	-	-
3672x1536 (2.39:1)	**3 680**	**1 536**	5 701 632	-	-	-	-	-	-	-
3840x2160 (4*HD)	**3 840**	**2 160**	8 355 840	-	-	-	-	-	-	-
4Kx2K	**4 096**	**2 048**	8 388 608	-	-	-	-	-	-	-
4096x2160	**4 096**	**2 160**	8 912 896	-	-	-	-	-	-	-
4096x2304 (16:9)	**4 096**	**2 304**	9 437 184	-	-	-	-	-	-	-
7680x4320	**7 680**	**4 320**	33 423 360	-	-	-	-	-	-	-
8192x4096	**8 192**	**4 096**	33 554 432	-	-	-	-	-	-	-
8192x4320	**8 192**	**4 320**	35 651 584	-	-	-	-	-	-	-

Table A.12 – Maximum picture rates (pictures per second) at level 5 to 6.2 for some example picture sizes when MinCbSizeY is equal to 64

Level:				5	5.1	5.2	6	6.1	6.2
Max luma picture size (samples):				8 912 896	8 912 896	8 912 896	35 651 584	35 651 584	35 651 584
Max luma sample rate (samples/sec)				267 386 880	534 773 760	1 069 547 520	1 069 547 520	2 139 095 040	4 278 190 080
Format nickname	Luma width	Luma height	Luma picture size						
SQCIF	128	96	16 384	300.0	300.0	300.0	300.0	300.0	300.0
QCIF	176	144	36 864	300.0	300.0	300.0	300.0	300.0	300.0
QVGA	320	240	81 920	300.0	300.0	300.0	300.0	300.0	300.0
525 SIF	352	240	98 304	300.0	300.0	300.0	300.0	300.0	300.0
CIF	352	288	122 880	300.0	300.0	300.0	300.0	300.0	300.0
525 HHR	352	480	196 608	300.0	300.0	300.0	300.0	300.0	300.0
625 HHR	352	576	221 184	300.0	300.0	300.0	300.0	300.0	300.0
Q720p	640	360	245 760	300.0	300.0	300.0	300.0	300.0	300.0
VGA	640	480	327 680	300.0	300.0	300.0	300.0	300.0	300.0
525 4SIF	704	480	360 448	300.0	300.0	300.0	300.0	300.0	300.0
525 SD	720	480	393 216	300.0	300.0	300.0	300.0	300.0	300.0
4CIF	704	576	405 504	300.0	300.0	300.0	300.0	300.0	300.0
625 SD	720	576	442 368	300.0	300.0	300.0	300.0	300.0	300.0
480p (16:9)	864	480	458 752	300.0	300.0	300.0	300.0	300.0	300.0
SVGA	800	600	532 480	300.0	300.0	300.0	300.0	300.0	300.0
QHD	960	540	552 960	300.0	300.0	300.0	300.0	300.0	300.0
XGA	1 024	768	786 432	300.0	300.0	300.0	300.0	300.0	300.0
720p HD	1 280	720	983 040	272.0	300.0	300.0	300.0	300.0	300.0
4VGA	1 280	960	1 228 800	217.6	300.0	300.0	300.0	300.0	300.0
SXGA	1 280	1 024	1 310 720	204.0	300.0	300.0	300.0	300.0	300.0
525 16SIF	1 408	960	1 351 680	197.8	300.0	300.0	300.0	300.0	300.0
16CIF	1 408	1 152	1 622 016	164.8	300.0	300.0	300.0	300.0	300.0
4SVGA	1 600	1 200	1 945 600	137.4	274.8	300.0	300.0	300.0	300.0
1080 HD	1 920	1 080	2 088 960	128.0	256.0	300.0	300.0	300.0	300.0
2Kx1K	2 048	1 024	2 097 152	127.5	255.0	300.0	300.0	300.0	300.0
2Kx1080	2 048	1 080	2 228 224	120.0	240.0	300.0	300.0	300.0	300.0
4XGA	2 048	1 536	3 145 728	85.0	170.0	300.0	300.0	300.0	300.0
16VGA	2 560	1 920	4 915 200	54.4	108.8	217.6	217.6	300.0	300.0
3616x1536 (2.35:1)	3 616	1 536	5 603 328	47.7	95.4	190.8	190.8	300.0	300.0
3672x1536 (2.39:1)	3 680	1 536	5 701 632	46.8	93.7	187.5	187.5	300.0	300.0
3840x2160 (4*HD)	3 840	2 160	8 355 840	32.0	64.0	128.0	128.0	256.0	300.0
4Kx2K	4 096	2 048	8 388 608	31.8	63.7	127.5	127.5	255.0	300.0
4096x2160	4 096	2 160	8 912 896	30.0	60.0	120.0	120.0	240.0	300.0
4096x2304 (16:9)	4 096	2 304	9 437 184	-	-	-	113.3	226.6	300.0
4096x3072	4 096	3 072	12 582 912	-	-	-	85.0	170.0	300.0
7680x4320	7 680	4 320	33 423 360	-	-	-	32.0	64.0	128.0
8192x4096	8 192	4 096	33 554 432	-	-	-	31.8	63.7	127.5
8192x4320	8 192	4 320	35 651 584	-	-	-	30.0	60.0	120.0

The following should be noted in regard to the examples shown in Tables A.11 and A.12:

– This is a variable-picture-size Specification. The specific listed picture sizes are illustrative examples only.

– The example luma picture sizes were computed by rounding up the luma width and luma height to multiples of 64 before computing the product of these quantities, to reflect the potential use of MinCbSizeY equal to 64 for these picture sizes, as pic_width_in_luma_samples and pic_height_in_luma_samples are each required to be a multiple of MinCbSizeY. For some illustrated values of luma width and luma height, a somewhat higher number of pictures per second can be supported when MinCbSizeY is less than 64.

– In cases where the maximum picture rate value is not an integer multiple of 0.1 pictures per second, the given maximum picture rate values have been rounded down to the largest integer multiple of 0.1 frames per second that does not exceed the exact value. For example, for level 3.1, the maximum picture rate for 720p HD has been rounded down to 33.7 from an exact value of 33.75.
– As used in the examples, "525" refers to typical use for environments using 525 analogue scan lines (of which approximately 480 lines contain the visible picture region) and "625" refers to environments using 625 analogue scan lines (of which approximately 576 lines contain the visible picture region).
– XGA is also known as (aka) XVGA, 4SVGA aka UXGA, 16XGA aka 4Kx3K, CIF aka 625 SIF, 625 HHR aka 2CIF aka half 625 D-1, aka half 625 ITU-R BT.601, 525 SD aka 525 D-1 aka 525 ITU-R BT.601, 625 SD aka 625 D-1 aka 625 ITU-R BT.601.

Annex B

Byte stream format

(This annex forms an integral part of this Recommendation | International Standard.)

B.1 General

This annex specifies syntax and semantics of a byte stream format specified for use by applications that deliver some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as Rec. ITU-T H.222.0 | ISO/IEC 13818-1 systems or Recommendation ITU-T H.320 systems. For bit-oriented delivery, the bit order for the byte stream format is specified to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

The byte stream format consists of a sequence of byte stream NAL unit syntax structures. Each byte stream NAL unit syntax structure contains one start code prefix followed by one nal_unit(NumBytesInNalUnit) syntax structure. It may (and under some circumstances, it shall) also contain an additional zero_byte syntax element. It may also contain one or more additional trailing_zero_8bits syntax elements. When it is the first byte stream NAL unit in the bitstream, it may also contain one or more additional leading_zero_8bits syntax elements.

B.2 Byte stream NAL unit syntax and semantics

B.2.1 Byte stream NAL unit syntax

byte_stream_nal_unit(NumBytesInNalUnit) {	**Descriptor**
while(next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)	
leading_zero_8bits /* equal to 0x00 */	f(8)
if(next_bits(24) != 0x000001)	
zero_byte /* equal to 0x00 */	f(8)
start_code_prefix_one_3bytes /* equal to 0x000001 */	f(24)
nal_unit(NumBytesInNalUnit)	
while(more_data_in_byte_stream( ) && next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)	
trailing_zero_8bits /* equal to 0x00 */	f(8)
}	

B.2.2 Byte stream NAL unit semantics

The order of byte stream NAL units in the byte stream shall follow the decoding order of the NAL units contained in the byte stream NAL units (see clause 7.4.2.4). The content of each byte stream NAL unit is associated with the same access unit as the NAL unit contained in the byte stream NAL unit (see clause 7.4.2.4.4).

leading_zero_8bits is a byte equal to 0x00.

NOTE – The leading_zero_8bits syntax element can only be present in the first byte stream NAL unit of the bitstream, because (as shown in the syntax diagram of clause B.2.1) any bytes equal to 0x00 that follow a NAL unit syntax structure and precede the four-byte sequence 0x00000001 (which is to be interpreted as a zero_byte followed by a start_code_prefix_one_3bytes) will be considered to be trailing_zero_8bits syntax elements that are part of the preceding byte stream NAL unit.

zero_byte is a single byte equal to 0x00.

When one or more of the following conditions are true, the zero_byte syntax element shall be present:

– The nal_unit_type within the nal_unit( ) syntax structure is equal to VPS_NUT, SPS_NUT or PPS_NUT.

– The byte stream NAL unit syntax structure contains the first NAL unit of an access unit in decoding order, as specified in clause 7.4.2.4.4.

start_code_prefix_one_3bytes is a fixed-value sequence of 3 bytes equal to 0x000001. This syntax element is called a start code prefix.

trailing_zero_8bits is a byte equal to 0x00.

B.3 Byte stream NAL unit decoding process

Input to this process consists of an ordered stream of bytes consisting of a sequence of byte stream NAL unit syntax structures.

Output of this process consists of a sequence of NAL unit syntax structures.

At the beginning of the decoding process, the decoder initializes its current position in the byte stream to the beginning of the byte stream. It then extracts and discards each leading_zero_8bits syntax element (when present), moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next four bytes in the bitstream form the four-byte sequence 0x00000001.

The decoder then performs the following step-wise process repeatedly to extract and decode each NAL unit syntax structure in the byte stream until the end of the byte stream has been encountered (as determined by unspecified means) and the last NAL unit in the byte stream has been decoded:

1. When the next four bytes in the bitstream form the four-byte sequence 0x00000001, the next byte in the byte stream (which is a zero_byte syntax element) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this discarded byte.

2. The next three-byte sequence in the byte stream (which is a start_code_prefix_one_3bytes) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this three-byte sequence.

3. NumBytesInNalUnit is set equal to the number of bytes starting with the byte at the current position in the byte stream up to and including the last byte that precedes the location of one or more of the following conditions:
   - A subsequent byte-aligned three-byte sequence equal to 0x000000,
   - A subsequent byte-aligned three-byte sequence equal to 0x000001,
   - The end of the byte stream, as determined by unspecified means.

4. NumBytesInNalUnit bytes are removed from the bitstream and the current position in the byte stream is advanced by NumBytesInNalUnit bytes. This sequence of bytes is nal_unit(NumBytesInNalUnit) and is decoded using the NAL unit decoding process.

5. When the current position in the byte stream is not at the end of the byte stream (as determined by unspecified means) and the next bytes in the byte stream do not start with a three-byte sequence equal to 0x000001 and the next bytes in the byte stream do not start with a four byte sequence equal to 0x00000001, the decoder extracts and discards each trailing_zero_8bits syntax element, moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next bytes in the byte stream form the four-byte sequence 0x00000001 or the end of the byte stream has been encountered (as determined by unspecified means).

B.4 Decoder byte-alignment recovery (informative)

This clause does not form an integral part of this Specification.

Many applications provide data to a decoder in a manner that is inherently byte aligned, and thus have no need for the bit-oriented byte alignment detection procedure described in this clause.

A decoder is said to have byte alignment with a bitstream when the decoder has determined whether or not the positions of data in the bitstream are byte-aligned. When a decoder does not have byte alignment with the bitstream, the decoder may examine the incoming bitstream for the binary pattern '00000000 00000000 00000000 00000001' (31 consecutive bits equal to 0 followed by a bit equal to 1). The bit immediately following this pattern is the first bit of an aligned byte following a start code prefix. Upon detecting this pattern, the decoder will be byte-aligned with the bitstream and positioned at the start of a NAL unit in the bitstream.

Once byte aligned with the bitstream, the decoder can examine the incoming bitstream data for subsequent three-byte sequences 0x000001 and 0x000003.

When the three-byte sequence 0x000001 is detected, this is a start code prefix.

When the three-byte sequence 0x000003 is detected, the third byte (0x03) is an emulation_prevention_three_byte to be discarded as specified in clause 7.4.2.

When an error in the bitstream syntax is detected (e.g., a non-zero value of the forbidden_zero_bit or one of the three-byte or four-byte sequences that are prohibited in clause 7.4.2), the decoder may consider the detected condition as an indication that byte alignment may have been lost and may discard all bitstream data until the detection of byte alignment at a later position in the bitstream as described above in this clause.

Annex C

Hypothetical reference decoder

(This annex forms an integral part of this Recommendation | International Standard.)

C.1 General

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

Two types of bitstreams or bitstream subsets are subject to HRD conformance checking for this Specification. The first type, called a Type I bitstream, is a NAL unit stream containing only the VCL NAL units and NAL units with nal_unit_type equal to FD_NUT (filler data NAL units) for all access units in the bitstream. The second type, called a Type II bitstream, contains, in addition to the VCL NAL units and filler data NAL units for all access units in the bitstream, at least one of the following:

– additional non-VCL NAL units other than filler data NAL units,

– all leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes and trailing_zero_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B).

NOTE 1 – Decoders conforming to profiles specified in Annex A do not use NAL units with nuh_layer_id greater than 0 (e.g., access unit delimiter NAL units with nuh_layer_id greater than 0) for access unit boundary detection, except for identification of whether a NAL unit is a VCL or non-VCL NAL unit. Consequently, hypothetical reference decoder (HRD) parameters carried in non-scalable-nested buffering period, picture timing and decoding unit information SEI messages apply to access units that are identified based on such access unit boundary detection.

Figure C.1 shows the types of bitstream conformance points checked by the HRD.

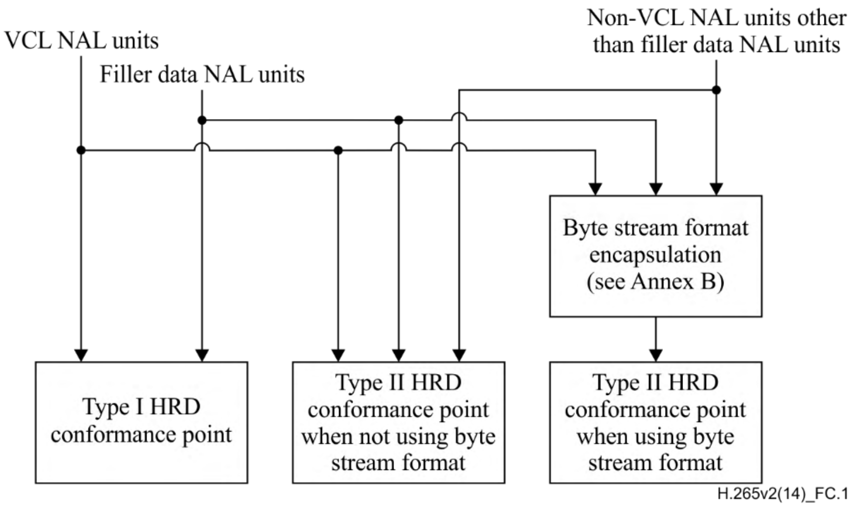


Figure C.1 – Structure of byte streams and NAL unit streams for HRD conformance checks

The syntax elements of non-VCL NAL units (or their default values for some of the syntax elements), required for the HRD, are specified in the semantic clauses of clause 7, Annexes D and E.

Two types of HRD parameter sets (NAL HRD parameters and VCL HRD parameters) are used. The HRD parameter sets are signalled through the hrd_parameters( ) syntax structure, which may be part of the SPS syntax structure or the VPS syntax structure.

Two sets of bitstream conformance tests are needed for checking the conformance of a bitstream, which is referred to as the entire bitstream, denoted as entireBitstream. The first set of bitstream conformance tests are for testing the conformance of the entire bitstream and its temporal subsets, regardless of whether there is a layer set specified by the active VPS that contains all the nuh_layer_id values of VCL NAL units present in the entire bitstream. The second set of bitstream conformance tests are for testing the conformance of the layer sets specified by the active VPS and their temporal subsets. For all these tests, only the base layer pictures (i.e., pictures with nuh_layer_id equal to 0) are decoded and other pictures are ignored by the decoder when the decoding process is invoked.

For each test, the following ordered steps apply in the order listed, followed by the processes described after these steps in this clause:

1. An operation point under test, denoted as TargetOp, is selected by selecting a layer identifier list OpLayerIdList and a target highest TemporalId value OpTid. The layer identifier list OpLayerIdList of TargetOp either consists of all the nuh_layer_id values of the VCL NAL units present in entireBitstream or consists of all the nuh_layer_id values of a layer set specified by the active VPS. The value of OpTid is in the range of 0 to sps_max_sub_layers_minus1, inclusive. The values of OpLayerIdList and OpTid are such that the sub-bitstream BitstreamToDecode that is the output by invoking the sub-bitstream extraction process as specified in clause 10 with entireBitstream, OpTid and OpLayerIdList as inputs satisfy both of the following conditions:
   – There is at least one VCL NAL unit in BitstreamToDecode with nuh_layer_id equal to each of the nuh_layer_id values in OpLayerIdList.
   – There is at least one VCL NAL unit with TemporalId equal to OpTid in BitstreamToDecode.
2. TargetDecLayerIdList is set equal to OpLayerIdList of TargetOp and HighestTid is set equal to OpTid of TargetOp.
3. The hrd_parameters( ) syntax structure and the sub_layer_hrd_parameters( ) syntax structure applicable to TargetOp are selected. If TargetDecLayerIdList contains all nuh_layer_id values present in entireBitstream, the hrd_parameters( ) syntax structure in the active SPS (or provided through an external means not specified in this Specification) is selected. Otherwise, the hrd_parameters( ) syntax structure in the active VPS (or provided through some external means not specified in this Specification) that applies to TargetOp is selected. Within the selected hrd_parameters( ) syntax structure, if BitstreamToDecode is a Type I bitstream, the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows the condition "if(vcl_hrd_parameters_present_flag)" is selected and the variable NalHrdModeFlag is set equal to 0; otherwise (BitstreamToDecode is a Type II bitstream), the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows either the condition "if(vcl_hrd_parameters_present_flag)" (in this case the variable NalHrdModeFlag is set equal to 0) or the condition "if(nal_hrd_parameters_present_flag)" (in this case the variable NalHrdModeFlag is set equal to 1) is selected. When BitstreamToDecode is a Type II bitstream and NalHrdModeFlag is equal to 0, all non-VCL NAL units except filler data NAL units, and all leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes and trailing_zero_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B), when present, are discarded from BitstreamToDecode and the remaining bitstream is assigned to BitstreamToDecode.
4. An access unit associated with a buffering period SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) applicable to TargetOp is selected as the HRD initialization point and referred to as access unit 0. If TargetDecLayerIdList contains all nuh_layer_id values present in entireBitstream, the assocciated buffering period SEI message shall be either a non-scalable-nested SEI message or provided by external means. Otherwise, the assocciated buffering period SEI message shall be either a scalable-nested SEI message or provided by external means.
5. When sub_pic_hrd_params_present_flag in the selected hrd_parameters( ) syntax structure is equal to 1, the CPB is scheduled to operate either at the access unit level (in which case the variable SubPicHrdFlag is set equal to 0) or at the sub-picture level (in which case the variable SubPicHrdFlag is set equal to 1). Otherwise, SubPicHrdFlag is set equal to 0 and the CPB is scheduled to operate at the partition unit level.
6. For each access unit in BitstreamToDecode starting from access unit 0, the buffering period SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp is selected, the picture timing SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp is selected, and when SubPicHrdFlag is equal to 1 and sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 0, the decoding unit information SEI messages (present in BitstreamToDecode or available through external means not specified in this Specification) that are associated with decoding units in the access unit and apply to TargetOp are selected. If TargetDecLayerIdList contains all nuh_layer_id values present in entireBitstream, the selected buffering period, picture timing and decoding unit information SEI messages shall be either non-scalable-nested SEI messages or provided by external means. Otherwise, the selected buffering period, picture timing and decoding unit information SEI messages shall be either scalable-nested SEI messages or provided by external means.
7. A value of SchedSelIdx is selected. The selected SchedSelIdx shall be in the range of 0 to cpb_cnt_minus1[HighestTid], inclusive, where cpb_cnt_minus1[HighestTid] is found in the hrd_parameters( ) syntax structure as selected above.
8. When the coded picture in access unit 0 has nal_unit_type equal to CRA_NUT or BLA_W_LP and irap_cpb_params_present_flag in the selected buffering period SEI message is equal to 1, either of the following applies for selection of the initial CPB removal delay and delay offset:
   – If NalHrdModeFlag is equal to 1, the default initial CPB removal delay and delay offset represented by nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. Otherwise, the default initial CPB

removal delay and delay offset represented by vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. The variable DefaultInitCpbParamsFlag is set equal to 1.

– If NalHrdModeFlag is equal to 1, the alternative initial CPB removal delay and delay offset represented by nal_initial_alt_cpb_removal_delay[SchedSelIdx] and nal_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. Otherwise, the alternative initial CPB removal delay and delay offset represented by vcl_initial_alt_cpb_removal_delay[SchedSelIdx] and vcl_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. The variable DefaultInitCpbParamsFlag is set equal to 0, and the RASL access units associated with access unit 0 are discarded from BitstreamToDecode and the remaining bitstream is assigned to BitstreamToDecode.

Each conformance test consists of a combination of one option in each of the above steps. When there is more than one option for a step, for any particular conformance test only one option is chosen. All possible combinations of all the steps form the entire set of conformance tests. For each operation point under test, the number of bitstream conformance tests to be performed is equal to n0 * n1 * (n2 * 2 + n3) * n4, where the values of n0, n1, n2, n3 and n4 are specified as follows:

- n0 is derived as follows:
  - If BitstreamToDecode is a Type I bitstream, n0 is equal to 1.
  - Otherwise (BitstreamToDecode is a Type II bitstream), n0 is equal to 2.
- n1 is equal to cpb_cnt_minus1[HighestTid] + 1.
- n2 is the number of access units in BitstreamToDecode that each is associated with a buffering period SEI message applicable to TargetOp and for each of which both of the following conditions are true:
  - nal_unit_type is equal to CRA_NUT or BLA_W_LP for the VCL NAL units.
  - The associated buffering period SEI message applicable to TargetOp has irap_cpb_params_present_flag equal to 1.
- n3 is the number of access units in BitstreamToDecode that each is associated with a buffering period SEI message applicable to TargetOp and for each of which one or both of the following conditions are true:
  - nal_unit_type is equal to neither CRA_NUT nor BLA_W_LP for the VCL NAL units.
  - The associated buffering period SEI message applicable to TargetOp has irap_cpb_params_present_flag equal to 0.
- n4 is derived as follows:
  - If sub_pic_hrd_params_present_flag in the selected hrd_parameters( ) syntax structure is equal to 0, n4 is equal to 1.
  - Otherwise, n4 is equal to 2.

When BitstreamToDecode is a Type II bitstream, the following applies:

- If the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows the condition "if(vcl_hrd_parameters_present_flag)" is selected, the test is conducted at the Type I conformance point shown in Figure C.1, and only VCL and filler data NAL units are counted for the input bit rate and CPB storage.
- Otherwise (the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows the condition "if(nal_hrd_parameters_present_flag)" is selected), the test is conducted at the Type II conformance point shown in Figure C.1, and all bytes of the Type II bitstream, which may be a NAL unit stream or a byte stream, are counted for the input bit rate and CPB storage.

  NOTE 2 – NAL HRD parameters established by a value of SchedSelIdx for the Type II conformance point shown in Figure C.1 are sufficient to also establish VCL HRD conformance for the Type I conformance point shown in Figure C.1 for the same values of InitCpbRemovalDelay[SchedSelIdx], BitRate[SchedSelIdx] and CpbSize[SchedSelIdx] for the variable bit rate (VBR) case (cbr_flag[SchedSelIdx] equal to 0). This is because the data flow into the Type I conformance point is a subset of the data flow into the Type II conformance point and because, for the VBR case, the CPB is allowed to become empty and stay empty until the time a next picture is scheduled to begin to arrive. For example, when decoding a CVS conforming to one or more of the profiles specified in Annex A using the decoding process specified in clauses 2 through 10, when NAL HRD parameters are provided for the Type II conformance point that not only fall within the bounds set for NAL HRD parameters for profile conformance in item f) of clause A.4.2 but also fall within the bounds set for VCL HRD parameters for profile conformance in item e) of clause A.4.2, conformance of the VCL HRD for the Type I conformance point is also assured to fall within the bounds of item e) of clause A.4.2.

All VPSs, SPSs and PPSs referred to in the VCL NAL units and the corresponding buffering period, picture timing and decoding unit information SEI messages shall be conveyed to the HRD, in a timely manner, either in the bitstream (by non-VCL NAL units), or by other means not specified in this Specification.

In Annexes C, D and E, the specification for "presence" of non-VCL NAL units that contain VPSs, SPSs, PPSs, buffering period SEI messages, picture timing SEI messages or decoding unit information SEI messages is also satisfied when those

NAL units (or just some of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

NOTE 3 – As an example, synchronization of such a non-VCL NAL unit, conveyed by means other than presence in the bitstream, with the NAL units that are present in the bitstream, can be achieved by indicating two points in the bitstream, between which the non-VCL NAL unit would have been present in the bitstream, had the encoder decided to convey it in the bitstream.

When the content of such a non-VCL NAL unit is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the non-VCL NAL unit is not required to use the same syntax as specified in this Specification.

NOTE 4 – When HRD information is contained within the bitstream, it is possible to verify the conformance of a bitstream to the requirements of this clause based solely on information contained in the bitstream. When the HRD information is not present in the bitstream, as is the case for all "stand-alone" Type I bitstreams, conformance can only be verified when the HRD data are supplied by some other means not specified in this Specification.

The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C.2.

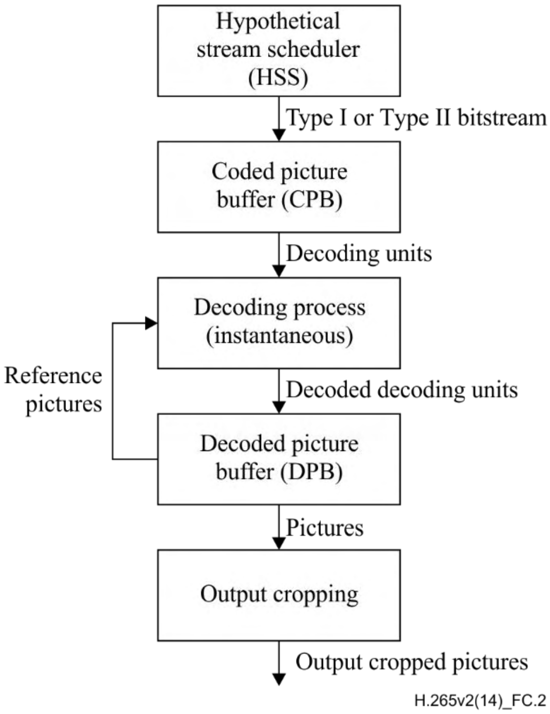


Figure C.2 – HRD buffer model

For each bitstream conformance test, the CPB size (number of bits) is CpbSize[SchedSelIdx] as specified in clause E.3.3, where SchedSelIdx and the HRD parameters are specified above in this clause. The DPB size (number of picture storage buffers) is sps_max_dec_pic_buffering_minus1[HighestTid] + 1.

If SubPicHrdFlag is equal to 0, the HRD operates at access unit level and each decoding unit is an access unit. Otherwise the HRD operates at sub-picture level and each decoding unit is a subset of an access unit.

NOTE 5 – If the HRD operates at access unit level, each time when some bits are removed from the CPB, a decoding unit that is an entire access unit is removed from the CPB. Otherwise (the HRD operates at sub-picture level), each time when some bits are removed from the CPB, a decoding unit that is a subset of an access unit is removed from the CPB. Regardless of whether the HRD operates at access unt level or sub-picture level, each time when some picture is output from the DPB, an entire decoded picture is output from the DPB, though the picture output time is derived based on the differently derived CPB removal times and the differently signalled DPB output delays.

The following is specified for expressing the constraints in this annex:

– Each access unit is referred to as access unit n, where the number n identifies the particular access unit. Access unit 0 is selected per step 4 above. The value of n is incremented by 1 for each subsequent access unit in decoding order.

– Each decoding unit is referred to as decoding unit m, where the number m identifies the particular decoding unit. The first decoding unit in decoding order in access unit 0 is referred to as decoding unit 0. The value of m is incremented by 1 for each subsequent decoding unit in decoding order.

NOTE 6 – The numbering of decoding units is relative to the first decoding unit in access unit 0.

– Picture n refers to the coded picture or the decoded picture of access unit n.

The HRD operates as follows:

– The HRD is initialized at decoding unit 0, with both the CPB and the DPB being set to be empty (the DPB fullness is set equal to 0).

NOTE 7 – After initialization, the HRD is not initialized again by subsequent buffering period SEI messages.

– Data associated with decoding units that flow into the CPB according to a specified arrival schedule are delivered by the hypothetical stream scheduler (HSS).

– The data associated with each decoding unit are removed and decoded instantaneously by the instantaneous decoding process at the CPB removal time of the decoding unit.

– Each decoded picture is placed in the DPB.

– A decoded picture is removed from the DPB when it becomes no longer needed for inter prediction reference and no longer needed for output.

For each bitstream conformance test, the operation of the CPB is specified in clause C.2, the instantaneous decoder operation is specified in clauses 2 through 10, the operation of the DPB is specified in clause C.3 and the output cropping is specified in clauses C.3.3 and C.5.2.2.

HSS and HRD information concerning the number of enumerated delivery schedules and their associated bit rates and buffer sizes is specified in clauses E.2.2 and E.3.2. The HRD is initialized as specified by the buffering period SEI message specified in clauses D.2.2 and D.3.2. The removal timing of decoding units from the CPB and output timing of decoded pictures from the DPB is specified using information in picture timing SEI messages (specified in clauses D.2.3 and 0) or in decoding unit information SEI messages (specified in clauses D.2.22 and D.3.22). All timing information relating to a specific decoding unit shall arrive prior to the CPB removal time of the decoding unit.

The requirements for bitstream conformance are specified in clause C.4 and the HRD is used to check conformance of bitstreams as specified above in this clause and to check conformance of decoders as specified in clause C.5.

NOTE 8 – While conformance is guaranteed under the assumption that all picture-rates and clocks used to generate the bitstream match exactly the values signalled in the bitstream, in a real system each of these may vary from the signalled or specified value.

All the arithmetic in this annex is performed with real values, so that no rounding errors can propagate. For example, the number of bits in a CPB just prior to or after removal of a decoding unit is not necessarily an integer.

The variable ClockTick is derived as follows and is called a clock tick:

$$\text{ClockTick} = \text{vui\_num\_units\_in\_tick} \div \text{vui\_time\_scale} \quad \text{(C-1)}$$

The variable ClockSubTick is derived as follows and is called a clock sub-tick:

$$\text{ClockSubTick} = \text{ClockTick} \div (\text{ tick\_divisor\_minus2} + 2 \text{ }) \quad \text{(C-2)}$$

C.2 Operation of coded picture buffer

C.2.1 General

The specifications in this clause apply independently to each set of coded picture buffer (CPB) parameters that is present and to both the Type I and Type II conformance points shown in Figure C.1 and the set of CPB parameters is selected as specified in clause C.1.

C.2.2 Timing of decoding unit arrival

If SubPicHrdFlag is equal to 0, the variable subPicParamsFlag is set equal to 0 and the process specified in the remainder of this clause is invoked with a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n.

Otherwise (SubPicHrdFlag is equal to 1), the process specified in the remainder of this clause is first invoked with the variable subPicParamsFlag set equal to 0 and a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n, and then invoked with subPicParamsFlag set equal to 1 and a decoding unit being considered as a subset of an access unit, for derivation of the initial and final CPB arrival times for the decoding units in access unit n.

The variables InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are derived as follows:

– If one or more of the following conditions are true, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_alt_cpb_removal_delay[SchedSelIdx] and nal_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1 or vcl_initial_alt_cpb_removal_delay[SchedSelIdx] and vcl_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause C.1:
  – Access unit 0 is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_RADL or BLA_N_LP, and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1.
  – Access unit 0 is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_LP or is a CRA access unit, and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:
    – UseAltCpbParamsFlag for access unit 0 is equal to 1.
    – DefaultInitCpbParamsFlag is equal to 0.
  – The value of subPicParamsFlag is equal to 1.
– Otherwise, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause C.1.

The time at which the first bit of decoding unit m begins to enter the CPB is referred to as the initial arrival time initArrivalTime[m].

The initial arrival time of decoding unit m is derived as follows:

– If the decoding unit is decoding unit 0 (i.e., when m is equal to 0), initArrivalTime[0] is set equal to 0.
– Otherwise (the decoding unit is decoding unit m with m > 0), the following applies:
  – If cbr_flag[SchedSelIdx] is equal to 1, the initial arrival time for decoding unit m is equal to the final arrival time (which is derived below) of decoding unit m − 1, i.e.,

```
if( !subPicParamsFlag )
    initArrivalTime[ m ] = AuFinalArrivalTime[ m − 1 ]                    (C-3)
else
    initArrivalTime[ m ] = DuFinalArrivalTime[ m − 1 ]
```

  – Otherwise (cbr_flag[SchedSelIdx] is equal to 0), the initial arrival time for decoding unit m is derived as follows:

```
if( !subPicParamsFlag )
    initArrivalTime[ m ] = Max( AuFinalArrivalTime[ m − 1 ], initArrivalEarliestTime[ m ] )
    (C-4)
else
    initArrivalTime[ m ] = Max( DuFinalArrivalTime[ m − 1 ], initArrivalEarliestTime[ m ] )
```

  where initArrivalEarliestTime[m] is derived as follows:
    – The variable tmpNominalRemovalTime is derived as follows:

```
if( !subPicParamsFlag )
  tmpNominalRemovalTime = AuNominalRemovalTime[ m ]                    (C-5)
else
  tmpNominalRemovalTime = DuNominalRemovalTime[ m ]
```

    where AuNominalRemovalTime[m] and DuNominalRemovalTime[m] are the nominal CPB removal time of access unit m and decoding unit m, respectively, as specified in clause C.2.3.
    – If decoding unit m is not the first decoding unit of a subsequent buffering period, initArrivalEarliestTime[m] is derived as follows:

$$\text{initArrivalEarliestTime[m]} = \text{tmpNominalRemovalTime} - (\text{ InitCpbRemovalDelay[SchedSelIdx]} + \text{InitCpbRemovalDelayOffset[SchedSelIdx])} \div 90\,000 \quad \text{(C-6)}$$

– Otherwise (decoding unit m is the first decoding unit of a subsequent buffering period), initArrivalEarliestTime[m] is derived as follows:

$$\text{initArrivalEarliestTime[m]} = \text{tmpNominalRemovalTime} - (\text{ InitCpbRemovalDelay[SchedSelIdx]} \div 90\,000 \text{)} \quad \text{(C-7)}$$

The final arrival time for decoding unit m is derived as follows:

```
if( !subPicParamsFlag )
   AuFinalArrivalTime[ m ] = initArrivalTime[ m ] + sizeInbits[ m ] ÷ BitRate[ SchedSelIdx ] (C-8)
else
   DuFinalArrivalTime[ m ] = initArrivalTime[ m ] + sizeInbits[ m ] ÷ BitRate[ SchedSelIdx ]
```

where sizeInbits[m] is the size in bits of decoding unit m, counting the bits of the VCL NAL units and the filler data NAL units for the Type I conformance point or all bits of the Type II bitstream for the Type II conformance point, where the Type I and Type II conformance points are as shown in Figure C.1.

The values of SchedSelIdx, BitRate[SchedSelIdx] and CpbSize[SchedSelIdx] are constrained as follows:

– If the content of the selected hrd_parameters( ) syntax structures for the access unit containing decoding unit m and the previous access unit differ, the HSS selects a value SchedSelIdx1 of SchedSelIdx from among the values of SchedSelIdx provided in the selected hrd_parameters( ) syntax structures for the access unit containing decoding unit m that results in a BitRate[SchedSelIdx1] or CpbSize[SchedSelIdx1] for the access unit containing decoding unit m. The value of BitRate[SchedSelIdx1] or CpbSize[SchedSelIdx1] may differ from the value of BitRate[SchedSelIdx0] or CpbSize[SchedSelIdx0] for the value SchedSelIdx0 of SchedSelIdx that was in use for the previous access unit.

– Otherwise, the HSS continues to operate with the previous values of SchedSelIdx, BitRate[SchedSelIdx] and CpbSize[SchedSelIdx].

When the HSS selects values of BitRate[SchedSelIdx] or CpbSize[SchedSelIdx] that differ from those of the previous access unit, the following applies:

– The variable BitRate[SchedSelIdx] comes into effect at the initial CPB arrival time of the current access unit.

– The variable CpbSize[SchedSelIdx] comes into effect as follows:

  – If the new value of CpbSize[SchedSelIdx] is greater than the old CPB size, it comes into effect at the initial CPB arrival time of the current access unit.

  – Otherwise, the new value of CpbSize[SchedSelIdx] comes into effect at the CPB removal time of the current access unit.

C.2.3 Timing of decoding unit removal and decoding of decoding unit

The variables InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are updated, and the variables CpbDelayOffset and DpbDelayOffset are derived, as follows:

– If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb_delay_offset, DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb_delay_offset, and InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_alt_cpb_removal_delay[SchedSelIdx] and nal_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_alt_cpb_removal_delay[SchedSelIdx] and vcl_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause C.1:

  – Access unit 0 is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_RADL or BLA_N_LP and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1.

  – Access unit 0 is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_LP or is a CRA access unit and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:

– UseAltCpbParamsFlag for access unit 0 is equal to 1.
– DefaultInitCpbParamsFlag is equal to 0.

– Otherwise, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause C.1, CpbDelayOffset and DpbDelayOffset are both set equal to 0.

The nominal removal time of the access unit n from the CPB is specified as follows:

– If access unit n is the access unit with n equal to 0 (the access unit that initializes the HRD), the nominal removal time of the access unit from the CPB is specified by:

$$\text{AuNominalRemovalTime[0]} = \text{InitCpbRemovalDelay[SchedSelIdx]} \div 90\,000 \quad \text{(C-9)}$$

– Otherwise, the following applies:

  – When access unit n is the first access unit of a buffering period that does not initialize the HRD, the following applies:

  The nominal removal time of the access unit n from the CPB is specified by:

```
if( !concatenationFlag ) {
    baseTime = AuNominalRemovalTime[ firstPicInPrevBuffPeriod ]
    tmpCpbRemovalDelay = AuCpbRemovalDelayVal
    tmpCpbDelayOffset = CpbDelayOffset
} else {
    baseTime1 = AuNominalRemovalTime[ prevNonDiscardablePic ]
    tmpCpbRemovalDelay1 = ( auCpbRemovalDelayDeltaMinus1 + 1 )
    baseTime2 = AuNominalRemovalTime[ n − 1 ]                                   (C-10)
    tmpCpbRemovalDelay2 = Ceil( ( InitCpbRemovalDelay[ SchedSelIdx ] ÷ 90 000 +

    AuFinalArrivalTime[ n − 1 ] − AuNominalRemovalTime[ n − 1 ] ) ÷ ClockTick ) )
    if( baseTime1 + ClockTick * tmpCpbRemovalDelay1 <
            baseTime2 + ClockTick * tmpCpbRemovalDelay2 ) {
        baseTime = baseTime2
        tmpCpbRemovalDelay = tmpCpbRemovalDelay2
    } else {
        baseTime = baseTime1
        tmpCpbRemovalDelay = tmpCpbRemovalDelay1
    }
    tmpCpbDelayOffset = 0
}
AuNominalRemovalTime[ n ] = baseTime +
                            ClockTick * ( tmpCpbRemovalDelay − tmpCpbDelayOffset )
```

  where AuNominalRemovalTime[firstPicInPrevBuffPeriod] is the nominal removal time of the first access unit of the previous buffering period, AuNominalRemovalTime[prevNonDiscardablePic] is the nominal removal time of the preceding picture in decoding order with TemporalId equal to 0 that is not a RASL, RADL or SLNR picture, AuCpbRemovalDelayVal is the value of AuCpbRemovalDelayVal derived according to au_cpb_removal_delay_minus1 in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n and concatenationFlag and auCpbRemovalDelayDeltaMinus1 are the values of the syntax elements concatenation_flag and au_cpb_removal_delay_delta_minus1, respectively, in the buffering period SEI message, selected as specified in clause C.1, associated with access unit n.

  After the derivation of the nominal CPB removal time and before the derivation of the DPB output time of access unit n, the values of CpbDelayOffset and DpbDelayOffset are updated as follows:

  – If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb_delay_offset, and DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb_delay_offset, where the buffering period SEI message containing the syntax elements is selected as specified in clause C.1:

    – Access unit n is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_RADL or BLA_N_LP and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1.

    – Access unit n is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_LP or is a CRA access unit and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and UseAltCpbParamsFlag for access unit n is equal to 1.

  – Otherwise, CpbDelayOffset and DpbDelayOffset are both set equal to 0.

– When access unit n is not the first access unit of a buffering period, the nominal removal time of the access unit n from the CPB is specified by:

$$\begin{aligned}&\text{AuNominalRemovalTime[n]} = \text{AuNominalRemovalTime[firstPicInCurrBuffPeriod]} + \\ &\qquad \text{ClockTick} * (\ \text{AuCpbRemovalDelayVal} - \text{CpbDelayOffset}\ )\end{aligned} \tag{C-11}$$

where AuNominalRemovalTime[firstPicInCurrBuffPeriod] is the nominal removal time of the first access unit of the current buffering period and AuCpbRemovalDelayVal is the value of AuCpbRemovalDelayVal derived according to au_cpb_removal_delay_minus1 in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n.

When SubPicHrdFlag is equal to 1, the following applies:

– The variable duCpbRemovalDelayInc is derived as follows:

  – If sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 0, duCpbRemovalDelayInc is set equal to the value of du_spt_cpb_removal_delay_increment in the decoding unit information SEI message, selected as specified in clause C.1, associated with decoding unit m.

  – Otherwise, if du_common_cpb_removal_delay_flag is equal to 0, duCpbRemovalDelayInc is set equal to the value of du_cpb_removal_delay_increment_minus1[i] + 1 for decoding unit m in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n, where the value of i is 0 for the first num_nalus_in_du_minus1[0] + 1 consecutive NAL units in the access unit that contains decoding unit m, 1 for the subsequent num_nalus_in_du_minus1[1] + 1 NAL units in the same access unit, 2 for the subsequent num_nalus_in_du_minus1[2] + 1 NAL units in the same access unit, etc.

  – Otherwise, duCpbRemovalDelayInc is set equal to the value of du_common_cpb_removal_delay_increment_minus1 + 1 in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n.

– The nominal removal time of decoding unit m from the CPB is specified as follows, where AuNominalRemovalTime[n] is the nominal removal time of access unit n:

  – If decoding unit m is the last decoding unit in access unit n, the nominal removal time of decoding unit m DuNominalRemovalTime[m] is set equal to AuNominalRemovalTime[n].

  – Otherwise (decoding unit m is not the last decoding unit in access unit n), the nominal removal time of decoding unit m DuNominalRemovalTime[m] is derived as follows:

$$\begin{aligned}&\text{if(sub\_pic\_cpb\_params\_in\_pic\_timing\_sei\_flag)}\\ &\quad \text{DuNominalRemovalTime[m]} = \text{DuNominalRemovalTime[m + 1]} - \\ &\qquad \text{ClockSubTick} * \text{duCpbRemovalDelayInc}\\ &\text{else}\\ &\quad \text{DuNominalRemovalTime[m]} = \text{AuNominalRemovalTime[n]} - \\ &\qquad \text{ClockSubTick} * \text{duCpbRemovalDelayInc}\end{aligned} \tag{C-12}$$

If SubPicHrdFlag is equal to 0, the removal time of access unit n from the CPB is specified as follows, where AuFinalArrivalTime[n] and AuNominalRemovalTime[n] are the final CPB arrival time and nominal CPB removal time, respectively, of access unit n:

$$\begin{aligned}&\text{if(!low\_delay\_hrd\_flag[HighestTid] || AuNominalRemovalTime[n] >= AuFinalArrivalTime[n])}\\ &\quad \text{AuCpbRemovalTime[n]} = \text{AuNominalRemovalTime[n]}\\ &\text{else}\\ &\quad \text{AuCpbRemovalTime[n]} = \text{AuNominalRemovalTime[n]} + \text{ClockTick} * \\ &\qquad \text{Ceil((AuFinalArrivalTime[n]} - \text{AuNominalRemovalTime[n])} \div \text{ClockTick)}\end{aligned} \tag{C-13}$$

NOTE 1 – When low_delay_hrd_flag[HighestTid] is equal to 1 and AuNominalRemovalTime[n] is less than AuFinalArrivalTime[n], the size of access unit n is so large that it prevents removal at the nominal removal time.

Otherwise (SubPicHrdFlag is equal to 1), the removal time of decoding unit m from the CPB is specified as follows:

```
if( !low_delay_hrd_flag[ HighestTid ] || DuNominalRemovalTime[ m ] >= DuFinalArrivalTime[ m ]
)
    DuCpbRemovalTime[ m ] = DuNominalRemovalTime[ m ]
else                                                                    (C-14)
    DuCpbRemovalTime[ m ] = DuFinalArrivalTime[ m ]
```

NOTE 2 – When low_delay_hrd_flag[HighestTid] is equal to 1 and DuNominalRemovalTime[m] is less than DuFinalArrivalTime[m], the size of decoding unit m is so large that it prevents removal at the nominal removal time.

If SubPicHrdFlag is equal to 0, at the CPB removal time of access unit n, the access unit is instantaneously decoded.

Otherwise (SubPicHrdFlag is equal to 1), at the CPB removal time of decoding unit m, the decoding unit is instantaneously decoded, and when decoding unit m is the last decoding unit of access unit n, the following applies:

– Picture n is considered as decoded.
– The final CPB arrival time of access unit n, i.e., AuFinalArrivalTime[n], is set equal to the final CPB arrival time of the last decoding unit in access unit n, i.e., DuFinalArrivalTime[m].
– The nominal CPB removal time of access unit n, i.e., AuNominalRemovalTime[n], is set equal to the nominal CPB removal time of the last decoding unit in access unit n, i.e., DuNominalRemovalTime[m].
– The CPB removal time of access unit n, i.e., AuCpbRemovalTime[m], is set equal to the CPB removal time of the last decoding unit in access unit n, i.e., DuCpbRemovalTime[m].

C.3 Operation of the decoded picture buffer

C.3.1 General

The specifications in this clause apply independently to each set of decoded picture buffer (DPB) parameters selected as specified in clause C.1.

The decoded picture buffer contains picture storage buffers. Each of the picture storage buffers may contain a decoded picture that is marked as "used for reference" or is held for future output. The processes specified in clauses C.3.2, C.3.3, C.3.4 and C.3.5 are sequentially applied as specified below.

C.3.2 Removal of pictures from the DPB before decoding of the current picture

The removal of pictures from the DPB before decoding of the current picture (but after parsing the slice header of the first slice of the current picture) happens instantaneously at the CPB removal time of the first decoding unit of access unit n (containing the current picture) and proceeds as follows:

– The decoding process for RPS as specified in clause 8.3.2 is invoked.
– When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 that is not picture 0, the following ordered steps are applied:
  1. The variable NoOutputOfPriorPicsFlag is derived for the decoder under test as follows:
     – If the current picture is a CRA picture, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no_output_of_prior_pics_flag).
     – Otherwise, if the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 or sps_max_dec_pic_buffering_minus1[HighestTid] derived from the active SPS is different from the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 or sps_max_dec_pic_buffering_minus1[HighestTid], respectively, derived from the SPS active for the preceding picture, NoOutputOfPriorPicsFlag may (but should not) be set to 1 by the decoder under test, regardless of the value of no_output_of_prior_pics_flag.

       NOTE – Although setting NoOutputOfPriorPicsFlag equal to no_output_of_prior_pics_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.
     – Otherwise, NoOutputOfPriorPicsFlag is set equal to no_output_of_prior_pics_flag.

2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD, such that when the value of NoOutputOfPriorPicsFlag is equal to 1, all picture storage buffers in the DPB are emptied without output of the pictures they contain, and the DPB fullness is set equal to 0.

– When both of the following conditions are true for any pictures k in the DPB, all such pictures k in the DPB are removed from the DPB:
  – picture k is marked as "unused for reference".
  – picture k has PicOutputFlag equal to 0 or its DPB output time is less than or equal to the CPB removal time of the first decoding unit (denoted as decoding unit m) of the current picture n; i.e., DpbOutputTime[k] is less than or equal to DuCpbRemovalTime[m].
– For each picture that is removed from the DPB, the DPB fullness is decremented by one.

C.3.3 Picture output

The processes specified in this clause happen instantaneously at the CPB removal time of access unit n, AuCpbRemovalTime[n].

When picture n has PicOutputFlag equal to 1, its DPB output time DpbOutputTime[n] is derived as follows, where the variable firstPicInBufferingPeriodFlag is equal to 1 if access unit n is the first access unit of a buffering period and 0 otherwise:

```
if( !SubPicHrdFlag ) {
   DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockTick * picDpbOutputDelay          (C-15)
   if( firstPicInBufferingPeriodFlag )
      DpbOutputTime[ n ] −= ClockTick * DpbDelayOffset
} else
   DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockSubTick * picSptDpbOutputDuDelay
```

where picDpbOutputDelay is the value of pic_dpb_output_delay in the picture timing SEI message associated with access unit n, and picSptDpbOutputDuDelay is the value of pic_spt_dpb_output_du_delay, when present, in the decoding unit information SEI messages associated with access unit n, or the value of pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n when there is no decoding unit information SEI message associated with access unit n or no decoding unit information SEI message associated with access unit n has pic_spt_dpb_output_du_delay present.

NOTE – When the syntax element pic_spt_dpb_output_du_delay is not present in any decoding unit information SEI message associated with access unit n, the value is inferred to be equal to pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n.

The output of the current picture is specified as follows:

– If PicOutputFlag is equal to 1 and DpbOutputTime[n] is equal to AuCpbRemovalTime[n], the current picture is output.
– Otherwise, if PicOutputFlag is equal to 0, the current picture is not output, but will be stored in the DPB as specified in clause C.3.4.
– Otherwise (PicOutputFlag is equal to 1 and DpbOutputTime[n] is greater than AuCpbRemovalTime[n]), the current picture is output later and will be stored in the DPB (as specified in clause C.3.4) and is output at time DpbOutputTime[n] unless indicated not to be output by NoOutputOfPriorPicsFlag equal to 1.

When output, the picture is cropped, using the conformance cropping window specified in the active SPS for the picture.

When picture n is a picture that is output and is not the last picture of the bitstream that is output, the value of the variable DpbOutputInterval[n] is derived as follows:

$$\text{DpbOutputInterval[n]} = \text{DpbOutputTime[nextPicInOutputOrder]} - \text{DpbOutputTime[n]} \quad \text{(C-16)}$$

where nextPicInOutputOrder is the picture that follows picture n in output order and has PicOutputFlag equal to 1.

C.3.4 Current decoded picture marking and storage

The current decoded picture after the invocation of the in-loop filter process as specified in clause 8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one. When TwoVersionsOfCurrDecPicFlag is equal to 0 and pps_curr_pic_ref_enabled_flag is equal to 1, this picture is marked as "used for long-term reference". After all the slices of the current picture have been decoded, this picture is marked as "used for short-term reference".

When TwoVersionsOfCurrDecPicFlag is equal to 1, the current decoded picture before the invocation of the in-loop filter process as specified in clause 8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one, and this picture is marked as "used for long-term reference".

NOTE – Unless more memory than required by the level limit is available for storage of decoded pictures, decoders should start storing decoded parts of the current picture into the DPB when the first slice segment is decoded and continue storing more decoded samples as the decoding process proceeds.

C.3.5 Removal of pictures from the DPB after decoding of the current picture

When TwoVersionsOfCurrDecPicFlag is equal to 1, immediately after decoding of the current picture, at the CPB removal time of the last decoding unit of access unit n (containing the current picture), the current decoded picture before the invocation of the in-loop filter process as specified in clause 8.7 is removed from the DPB, and the DPB fullness is decremented by one.

C.4 Bitstream conformance

A bitstream of coded data conforming to this Specification shall fulfil all requirements specified in this clause.

The bitstream shall be constructed according to the syntax, semantics and constraints specified in this Specification outside of this annex.

The first coded picture in a bitstream shall be an IRAP picture, i.e., an IDR picture, a CRA picture or a BLA picture.

The bitstream is tested by the HRD for conformance as specified in clause C.1.

For each current picture, let the variables maxPicOrderCnt and minPicOrderCnt be set equal to the maximum and the minimum, respectively, of the PicOrderCntVal values of the following pictures:

– The current picture.

– The previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL, or SLNR picture.

– The short-term reference pictures in the RPS of the current picture.

– All pictures n that have PicOutputFlag equal to 1, AuCpbRemovalTime[n] less than AuCpbRemovalTime[currPic] and DpbOutputTime[n] greater than or equal to AuCpbRemovalTime[currPic], where currPic is the current picture.

All of the following conditions shall be fulfilled for each of the bitstream conformance tests:

1. For each access unit n, with n greater than 0, associated with a buffering period SEI message, let the variable deltaTime90k[n] be specified as follows:

$$\text{deltaTime90k[n]} = 90\,000 * (\text{ AuNominalRemovalTime[n]} - \text{AuFinalArrivalTime[n} - 1\text{])} \quad \text{(C-17)}$$

The value of InitCpbRemovalDelay[SchedSelIdx] is constrained as follows:

– If cbr_flag[SchedSelIdx] is equal to 0, the following condition shall be true:

$$\text{InitCpbRemovalDelay[SchedSelIdx]} <= \text{Ceil(deltaTime90k[n])} \quad \text{(C-18)}$$

– Otherwise (cbr_flag[SchedSelIdx] is equal to 1), the following condition shall be true:

$$\text{Floor(deltaTime90k[n])} <= \text{InitCpbRemovalDelay[SchedSelIdx]} <= \text{Ceil(deltaTime90k[n])} \quad \text{(C-19)}$$

NOTE 1 – The exact number of bits in the CPB at the removal time of each picture may depend on which buffering period SEI message is selected to initialize the HRD. Encoders must take this into account to ensure that all specified constraints must be obeyed regardless of which buffering period SEI message is selected to initialize the HRD, as the HRD may be initialized at any one of the buffering period SEI messages.

2. A CPB overflow is specified as the condition in which the total number of bits in the CPB is greater than the CPB size. The CPB shall never overflow.

3. When low_delay_hrd_flag[HighestTid] is equal to 0, the CPB shall never underflow. A CPB underflow is specified as follows:

– If SubHrdFlag is equal to 0, a CPB underflow is specified as the condition in which the nominal CPB removal time of access unit n AuNominalRemovalTime[n] is less than the final CPB arrival time of access unit n AuFinalArrivalTime[n] for at least one value of n.

– Otherwise (SubPicHrdFlag is equal to 1), a CPB underflow is specified as the condition in which the nominal CPB removal time of decoding unit m DuNominalRemovalTime[m] is less than the final CPB arrival time of decoding unit m DuFinalArrivalTime[m] for at least one value of m.

4. When SubPicHrdFlag is equal to 1, low_delay_hrd_flag[HighestTid] is equal to 1 and the nominal removal time of a decoding unit m of access unit n is less than the final CPB arrival time of decoding unit m (i.e., DuNominalRemovalTime[m] < DuFinalArrivalTime[m]), the nominal removal time of access unit n shall be less than the final CPB arrival time of access unit n (i.e., AuNominalRemovalTime[n] < AuFinalArrivalTime[n]).

5. The nominal removal times of pictures from the CPB (starting from the second picture in decoding order) shall satisfy the constraints on AuNominalRemovalTime[n] and AuCpbRemovalTime[n] expressed in clauses A.4.1 through A.4.2.

6. For each current picture, after invocation of the process for removal of pictures from the DPB as specified in clause C.3.2, the number of decoded pictures in the DPB, including all pictures n that are marked as "used for reference", or that have PicOutputFlag equal to 1 and AuCpbRemovalTime[n] less than AuCpbRemovalTime[currPic], where currPic is the current picture, shall be less than or equal to sps_max_dec_pic_buffering_minus1[HighestTid].

7. All reference pictures shall be present in the DPB when needed for prediction. Each picture that has PicOutputFlag equal to 1 shall be present in the DPB at its DPB output time unless it is removed from the DPB before its output time by one of the processes specified in clause C.3.

8. For each current picture that is not an IRAP picture with NoRaslOutputFlag equal to 1, the value of maxPicOrderCnt − minPicOrderCnt shall be less than MaxPicOrderCntLsb / 2.

9. The value of DpbOutputInterval[n] as given by Equation C-16, which is the difference between the output time of a picture and that of the first picture following it in output order and having PicOutputFlag equal to 1, shall satisfy the constraint expressed in clause A.4.1 for the profile, tier and level specified in the bitstream using the decoding process specified in clauses 2 through 10.

10. For each current picture, when sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 1, let tmpCpbRemovalDelaySum be derived as follows:

```
tmpCpbRemovalDelaySum = 0
for( i = 0; i < num_decoding_units_minus1; i++ )                                   (C-20)
    tmpCpbRemovalDelaySum += du_cpb_removal_delay_increment_minus1[ i ] + 1
```

The value of ClockSubTick * tmpCpbRemovalDelaySum shall be equal to the difference between the nominal CPB removal time of the current access unit and the nominal CPB removal time of the first decoding unit in the current access unit in decoding order.

11. For any two pictures m and n in the same CVS, when DpbOutputTime[m] is greater than DpbOutputTime[n], the PicOrderCntVal of picture m shall be greater than the PicOrderCntVal of picture n.

NOTE 2 – All pictures of an earlier CVS in decoding order that are output are output before any pictures of a later CVS in decoding order. Within any particular CVS, the pictures that are output are output in increasing PicOrderCntVal order.

C.5 Decoder conformance

C.5.1 General

A decoder conforming to this Specification shall fulfil all requirements specified in this clause.

A decoder claiming conformance to a specific profile, tier and level shall be able to successfully decode all bitstreams that conform to the bitstream conformance requirements specified in clause C.4, in the manner specified in Annex A, provided that all VPSs, SPSs and PPSs referred to in the VCL NAL units and appropriate buffering period, picture timing and decoding unit information SEI messages are conveyed to the decoder, in a timely manner, either in the bitstream (by non-VCL NAL units), or by external means not specified in this Specification.

When a bitstream contains syntax elements that have values that are specified as reserved and it is specified that decoders shall ignore values of the syntax elements or NAL units containing the syntax elements having the reserved values, and

the bitstream is otherwise conforming to this Specification, a conforming decoder shall decode the bitstream in the same manner as it would decode a conforming bitstream and shall ignore the syntax elements or the NAL units containing the syntax elements having the reserved values as specified.

There are two types of conformance that can be claimed by a decoder: output timing conformance and output order conformance.

To check conformance of a decoder, test bitstreams conforming to the claimed profile, tier and level, as specified in clause C.4 are delivered by a hypothetical stream scheduler (HSS) both to the HRD and to the decoder under test (DUT). All cropped decoded pictures output by the HRD shall also be output by the DUT, each cropped decoded picture output by the DUT shall be a picture with PicOutputFlag equal to 1, and, for each such cropped decoded picture output by the DUT, the values of all samples that are output shall be equal to the values of the samples produced by the specified decoding process.

For output timing decoder conformance, the HSS operates as described above, with delivery schedules selected only from the subset of values of SchedSelIdx for which the bit rate and CPB size are restricted as specified in Annex A for the specified profile, tier and level or with "interpolated" delivery schedules as specified below for which the bit rate and CPB size are restricted as specified in Annex A. The same delivery schedule is used for both the HRD and the DUT.

When the HRD parameters and the buffering period SEI messages are present with cpb_cnt_minus1[HighestTid] greater than 0, the decoder shall be capable of decoding the bitstream as delivered from the HSS operating using an "interpolated" delivery schedule specified as having peak bit rate r, CPB size c(r) and initial CPB removal delay (f(r) ÷ r) as follows:

$$\alpha = (r - \text{BitRate}[\text{SchedSelIdx} - 1]) \div (\text{BitRate}[\text{SchedSelIdx}] - \text{BitRate}[\text{SchedSelIdx} - 1]), \quad \text{(C-21)}$$

$$c(r) = \alpha * \text{CpbSize}[\text{SchedSelIdx}] + (1 - \alpha) * \text{CpbSize}[\text{SchedSelIdx} - 1], \quad \text{(C-22)}$$

$$f(r) = \alpha * \text{InitCpbRemovalDelay}[\text{SchedSelIdx}] * \text{BitRate}[\text{SchedSelIdx}] + (1 - \alpha) * \text{InitCpbRemovalDelay}[\text{SchedSelIdx} - 1] * \text{BitRate}[\text{SchedSelIdx} - 1] \quad \text{(C-23)}$$

for any SchedSelIdx > 0 and r such that BitRate[SchedSelIdx − 1] <= r <= BitRate[SchedSelIdx] such that r and c(r) are within the limits as specified in Annex A for the maximum bit rate and buffer size for the specified profile, tier and level.

NOTE 1 – InitCpbRemovalDelay[SchedSelIdx] can be different from one buffering period to another and have to be re-calculated.

For output timing decoder conformance, an HRD as described above is used and the timing (relative to the delivery time of the first bit) of picture output is the same for both the HRD and the DUT up to a fixed delay.

For output order decoder conformance, the following applies:

– The HSS delivers the bitstream BitstreamToDecode to the DUT "by demand" from the DUT, meaning that the HSS delivers bits (in decoding order) only when the DUT requires more bits to proceed with its processing.

  NOTE 2 – This means that for this test, the coded picture buffer of the DUT could be as small as the size of the largest decoding unit.

– A modified HRD as described below is used, and the HSS delivers the bitstream to the HRD by one of the schedules specified in the bitstream BitstreamToDecode such that the bit rate and CPB size are restricted as specified in Annex A. The order of pictures output shall be the same for both the HRD and the DUT.

– The HRD CPB size is given by CpbSize[SchedSelIdx] as specified in clause E.3.3, where SchedSelIdx and the HRD parameters are selected as specified in clause C.1. The DPB size is given by sps_max_dec_pic_buffering_minus1[HighestTid] + 1. Removal time from the CPB for the HRD is the final bit arrival time and decoding is immediate. The operation of the DPB of this HRD is as described in clauses C.5.2 through C.5.2.3.

C.5.2 Operation of the output order DPB

C.5.2.1 General

The decoded picture buffer contains picture storage buffers. Each of the picture storage buffers contains a decoded picture that is marked as "used for reference" or is held for future output. The process for output and removal of pictures from the DPB before decoding of the current picture as specified in clause C.5.2.2 is invoked, the invocation of the process for current decoded picture marking and storage as specified in clause C.3.4, further followed by the invocation of the process for removal of pictures from the DPB after decoding of the current picture as specified in clause C.3.5, and finally followed by the invocation of the process for additional bumping as specified in clause C.5.2.3. The "bumping" process is specified in clause C.5.2.4 and is invoked as specified in clauses C.5.2.2 and C.5.2.3.

C.5.2.2 Output and removal of pictures from the DPB

The output and removal of pictures from the DPB before the decoding of the current picture (but after parsing the slice header of the first slice of the current picture) happens instantaneously when the first decoding unit of the access unit containing the current picture is removed from the CPB and proceeds as follows:

– The decoding process for RPS as specified in clause 8.3.2 is invoked.

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 that is not picture 0, the following ordered steps are applied:

  1. The variable NoOutputOfPriorPicsFlag is derived for the decoder under test as follows:

     – If the current picture is a CRA picture, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no_output_of_prior_pics_flag).

     – Otherwise, if the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 or sps_max_dec_pic_buffering_minus1[HighestTid] derived from the active SPS is different from the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 or sps_max_dec_pic_buffering_minus1[HighestTid], respectively, derived from the SPS active for the preceding picture, NoOutputOfPriorPicsFlag may (but should not) be set to 1 by the decoder under test, regardless of the value of no_output_of_prior_pics_flag.

       NOTE – Although setting NoOutputOfPriorPicsFlag equal to no_output_of_prior_pics_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.

     – Otherwise, NoOutputOfPriorPicsFlag is set equal to no_output_of_prior_pics_flag.

  2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD as follows:

     – If NoOutputOfPriorPicsFlag is equal to 1, all picture storage buffers in the DPB are emptied without output of the pictures they contain and the DPB fullness is set equal to 0.

     – Otherwise (NoOutputOfPriorPicsFlag is equal to 0), all picture storage buffers containing a picture that is marked as "not needed for output" and "unused for reference" are emptied (without output) and all non-empty picture storage buffers in the DPB are emptied by repeatedly invoking the "bumping" process specified in clause C.5.2.4 and the DPB fullness is set equal to 0.

– Otherwise (the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1), all picture storage buffers containing a picture which are marked as "not needed for output" and "unused for reference" are emptied (without output). For each picture storage buffer that is emptied, the DPB fullness is decremented by one. When one or more of the following conditions are true, the "bumping" process specified in clause C.5.2.4 is invoked repeatedly while further decrementing the DPB fullness by one for each additional picture storage buffer that is emptied, until none of the following conditions are true:

  – The number of pictures in the DPB that are marked as "needed for output" is greater than sps_max_num_reorder_pics[HighestTid].

  – sps_max_latency_increase_plus1[HighestTid] is not equal to 0 and there is at least one picture in the DPB that is marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to SpsMaxLatencyPictures[HighestTid].

  – The number of pictures in the DPB is greater than or equal to sps_max_dec_pic_buffering_minus1[HighestTid] + 1 − TwoVersionsOfCurrDecPicFlag.

C.5.2.3 Additional bumping

The processes specified in this clause happen instantaneously when the last decoding unit of access unit n containing the current picture is removed from the CPB.

When the current picture has PicOutputFlag equal to 1, for each picture in the DPB that is marked as "needed for output" and follows the current picture in output order, the associated variable PicLatencyCount is set equal to PicLatencyCount + 1.

The following applies:

– If the current decoded picture has PicOutputFlag equal to 1, it is marked as "needed for output" and its associated variable PicLatencyCount is set equal to 0.

– Otherwise (the current decoded picture has PicOutputFlag equal to 0), it is marked as "not needed for output".

When one or more of the following conditions are true, the "bumping" process specified in clause C.5.2.4 is invoked repeatedly until none of the following conditions are true:

– The number of pictures in the DPB that are marked as "needed for output" is greater than sps_max_num_reorder_pics[HighestTid].

– sps_max_latency_increase_plus1[HighestTid] is not equal to 0 and there is at least one picture in the DPB that is marked as "needed for output" for which the associated variable PicLatencyCount that is greater than or equal to SpsMaxLatencyPictures[HighestTid].

C.5.2.4 "Bumping" process

The "bumping" process consists of the following ordered steps:

1. The picture that is first for output is selected as the one having the smallest value of PicOrderCntVal of all pictures in the DPB marked as "needed for output".
2. The picture is cropped, using the conformance cropping window specified in the active SPS for the picture, the cropped picture is output, and the picture is marked as "not needed for output".
3. When the picture storage buffer that included the picture that was cropped and output contains a picture marked as "unused for reference", the picture storage buffer is emptied.

NOTE – For any two pictures picA and picB that belong to the same CVS and are output by the "bumping process", when picA is output earlier than picB, the value of PicOrderCntVal of picA is less than the value of PicOrderCntVal of picB.

Annex D

Supplemental enhancement information

(This annex forms an integral part of this Recommendation | International Standard.)

D.1 General

This annex specifies syntax and semantics for SEI message payloads.

SEI messages assist in processes related to decoding, display or other purposes. However, SEI messages are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Specification (see Annex C and clause F.13 for the specification of conformance). Some SEI message information is required to check bitstream conformance and for output timing decoder conformance.

In clause C.5.2 and in clause F.13 including its subclauses, specification for presence of SEI messages are also satisfied when those messages (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. When present in the bitstream, SEI messages shall obey the syntax and semantics specified in clause 7.3.5 and this annex. When the content of an SEI message is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the SEI message is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

D.2 SEI payload syntax

D.2.1 General SEI message syntax

sei_payload(payloadType, payloadSize) {	**Descriptor**
if(nal_unit_type = = PREFIX_SEI_NUT)	
if(payloadType = = 0)	
buffering_period(payloadSize)	
else if(payloadType = = 1)	
pic_timing(payloadSize)	
else if(payloadType = = 2)	
pan_scan_rect(payloadSize)	
else if(payloadType = = 3)	
filler_payload(payloadSize)	
else if(payloadType = = 4)	
user_data_registered_itu_t_t35(payloadSize)	
else if(payloadType = = 5)	
user_data_unregistered(payloadSize)	
else if(payloadType = = 6)	
recovery_point(payloadSize)	
else if(payloadType = = 9)	
scene_info(payloadSize)	
else if(payloadType = = 15)	
picture_snapshot(payloadSize)	
else if(payloadType = = 16)	
progressive_refinement_segment_start(payloadSize)	
else if(payloadType = = 17)	
progressive_refinement_segment_end(payloadSize)	
else if(payloadType = = 19)	
film_grain_characteristics(payloadSize)	
else if(payloadType = = 22)	
post_filter_hint(payloadSize)	
else if(payloadType = = 23)	
tone_mapping_info(payloadSize)	
else if(payloadType = = 45)	

frame_packing_arrangement(payloadSize)	
else if(payloadType = = 47)	
display_orientation(payloadSize)	
else if(payloadType = = 56)	
green_metadata(payloadsize) /* specified in ISO/IEC 23001-11 */	
else if(payloadType = = 128)	
structure_of_pictures_info(payloadSize)	
else if(payloadType = = 129)	
active_parameter_sets(payloadSize)	
else if(payloadType = = 130)	
decoding_unit_info(payloadSize)	
else if(payloadType = = 131)	
temporal_sub_layer_zero_idx(payloadSize)	
else if(payloadType = = 133)	
scalable_nesting(payloadSize)	
else if(payloadType = = 134)	
region_refresh_info(payloadSize)	
else if(payloadType = = 135)	
no_display(payloadSize)	
else if(payloadType = = 136)	
time_code(payloadSize)	
else if(payloadType = = 137)	
mastering_display_colour_volume(payloadSize)	
else if(payloadType = = 138)	
segmented_rect_frame_packing_arrangement(payloadSize)	
else if(payloadType = = 139)	
temporal_motion_constrained_tile_sets(payloadSize)	
else if(payloadType = = 140)	
chroma_resampling_filter_hint(payloadSize)	
else if(payloadType = = 141)	
knee_function_info(payloadSize)	
else if(payloadType = = 142)	
colour_remapping_info(payloadSize)	
else if(payloadType = = 143)	
deinterlaced_field_identification(payloadSize)	
else if(payloadType = = 144)	
content_light_level_info(payloadSize)	
else if(payloadType = = 145)	
dependent_rap_indication(payloadSize)	
else if(payloadType = = 146)	
coded_region_completion(payloadSize)	
else if(payloadType = = 147)	
alternative_transfer_characteristics(payloadSize)	
else if(payloadType = = 148)	
ambient_viewing_environment(payloadSize)	
else if(payloadType = = 149)	
content_colour_volume(payloadSize)	
else if(payloadType = = 150)	
equirectangular_projection(payloadSize)	
else if(payloadType = = 151)	
cubemap_projection(payloadSize)	
else if(payloadType = = 152)	
fisheye_video_info(payloadSize)	

else if(payloadType = = 154)	
sphere_rotation(payloadSize)	
else if(payloadType = = 155)	
regionwise_packing(payloadSize)	
else if(payloadType = = 156)	
omni_viewport(payloadSize)	
else if(payloadType = = 157)	
regional_nesting(payloadSize)	
else if(payloadType = = 158)	
mcts_extraction_info_sets(payloadSize)	
else if(payloadType = = 159)	
mcts_extraction_info_nesting(payloadSize)	
else if(payloadType = = 160)	
layers_not_present(payloadSize) /* specified in Annex F */	
else if(payloadType = = 161)	
inter_layer_constrained_tile_sets(payloadSize) /* specified in Annex F */	
else if(payloadType = = 162)	
bsp_nesting(payloadSize) /* specified in Annex F */	
else if(payloadType = = 163)	
bsp_initial_arrival_time(payloadSize) /* specified in Annex F */	
else if(payloadType = = 164)	
sub_bitstream_property(payloadSize) /* specified in Annex F */	
else if(payloadType = = 165)	
alpha_channel_info(payloadSize) /* specified in Annex F */	
else if(payloadType = = 166)	
overlay_info(payloadSize) /* specified in Annex F */	
else if(payloadType = = 167)	
temporal_mv_prediction_constraints(payloadSize) /* specified in Annex F */	
else if(payloadType = = 168)	
frame_field_info(payloadSize) /* specified in Annex F */	
else if(payloadType = = 176)	
three_dimensional_reference_displays_info(payloadSize) /* specified in Annex G */	
else if(payloadType = = 177)	
depth_representation_info(payloadSize) /* specified in Annex G */	
else if(payloadType = = 178)	
multiview_scene_info(payloadSize) /* specified in Annex G */	
else if(payloadType = = 179)	
multiview_acquisition_info(payloadSize) /* specified in Annex G */	
else if(payloadType = = 180)	
multiview_view_position(payloadSize) /* specified in Annex G */	
else if(payloadType = = 181)	
alternative_depth_info(payloadSize) /* specified in Annex I */	
else if(payloadType = = 200)	
sei_manifest(payloadSize)	
else if(payloadType = = 201)	
sei_prefix_indication(payloadSize)	
else if(payloadType = = 202)	
annotated_regions(payloadSize)	
else if(payloadType = = 205)	
shutter_interval_info(payloadSize)	
else	
reserved_sei_message(payloadSize)	

else /* nal_unit_type = = SUFFIX_SEI_NUT */	
if(payloadType = = 3)	
filler_payload(payloadSize)	
else if(payloadType = = 4)	
user_data_registered_itu_t_t35(payloadSize)	
else if(payloadType = = 5)	
user_data_unregistered(payloadSize)	
else if(payloadType = = 17)	
progressive_refinement_segment_end(payloadSize)	
else if(payloadType = = 22)	
post_filter_hint(payloadSize)	
else if(payloadType = = 132)	
decoded_picture_hash(payloadSize)	
else if(payloadType = = 146)	
coded_region_completion(payloadSize)	
else	
reserved_sei_message(payloadSize)	
if(more_data_in_payload( )) {	
if(payload_extension_present( ))	
reserved_payload_extension_data	u(v)
payload_bit_equal_to_one /* equal to 1 */	f(1)
while(!byte_aligned( ))	
payload_bit_equal_to_zero /* equal to 0 */	f(1)
}	
}	

D.2.2 Buffering period SEI message syntax

buffering_period(payloadSize) {	**Descriptor**
bp_seq_parameter_set_id	ue(v)
if(!sub_pic_hrd_params_present_flag)	
irap_cpb_params_present_flag	u(1)
if(irap_cpb_params_present_flag) {	
cpb_delay_offset	u(v)
dpb_delay_offset	u(v)
}	
concatenation_flag	u(1)
au_cpb_removal_delay_delta_minus1	u(v)
if(NalHrdBpPresentFlag) {	
for(i = 0; i < CpbCnt; i++) {	
nal_initial_cpb_removal_delay[i]	u(v)
nal_initial_cpb_removal_offset[i]	u(v)
if(sub_pic_hrd_params_present_flag || irap_cpb_params_present_flag) {	
nal_initial_alt_cpb_removal_delay[i]	u(v)
nal_initial_alt_cpb_removal_offset[i]	u(v)
}	
}	
}	
if(VclHrdBpPresentFlag) {	
for(i = 0; i < CpbCnt; i++) {	
vcl_initial_cpb_removal_delay[i]	u(v)
vcl_initial_cpb_removal_offset[i]	u(v)
if(sub_pic_hrd_params_present_flag || irap_cpb_params_present_flag) {	
vcl_initial_alt_cpb_removal_delay[i]	u(v)
vcl_initial_alt_cpb_removal_offset[i]	u(v)
}	
}	
}	
if(more_data_in_payload( ))	
if(payload_extension_present( ))	
use_alt_cpb_params_flag	u(1)
}	

D.2.3 Picture timing SEI message syntax

pic_timing(payloadSize) {	**Descriptor**
if(frame_field_info_present_flag) {	
pic_struct	u(4)
source_scan_type	u(2)
duplicate_flag	u(1)
}	
if(CpbDpbDelaysPresentFlag) {	
au_cpb_removal_delay_minus1	u(v)
pic_dpb_output_delay	u(v)
if(sub_pic_hrd_params_present_flag)	
pic_dpb_output_du_delay	u(v)
if(sub_pic_hrd_params_present_flag && sub_pic_cpb_params_in_pic_timing_sei_flag) {	
num_decoding_units_minus1	ue(v)
du_common_cpb_removal_delay_flag	u(1)
if(du_common_cpb_removal_delay_flag)	
du_common_cpb_removal_delay_increment_minus1	u(v)
for(i = 0; i <= num_decoding_units_minus1; i++) {	
num_nalus_in_du_minus1[i]	ue(v)
if(!du_common_cpb_removal_delay_flag && i < num_decoding_units_minus1)	
du_cpb_removal_delay_increment_minus1[i]	u(v)
}	
}	
}	
}	

D.2.4 Pan-scan rectangle SEI message syntax

pan_scan_rect(payloadSize) {	**Descriptor**
pan_scan_rect_id	ue(v)
pan_scan_rect_cancel_flag	u(1)
if(!pan_scan_rect_cancel_flag) {	
pan_scan_cnt_minus1	ue(v)
for(i = 0; i <= pan_scan_cnt_minus1; i++) {	
pan_scan_rect_left_offset[i]	se(v)
pan_scan_rect_right_offset[i]	se(v)
pan_scan_rect_top_offset[i]	se(v)
pan_scan_rect_bottom_offset[i]	se(v)
}	
pan_scan_rect_persistence_flag	u(1)
}	
}	

D.2.5 Filler payload SEI message syntax

filler_payload(payloadSize) {	**Descriptor**
for(k = 0; k < payloadSize; k++)	
ff_byte /* equal to 0xFF */	f(8)
}	

D.2.6 User data registered by Recommendation ITU-T T.35 SEI message syntax

user_data_registered_itu_t_t35(payloadSize) {	**Descriptor**
itu_t_t35_country_code	b(8)
if(itu_t_t35_country_code != 0xFF)	
i = 1	
else {	
itu_t_t35_country_code_extension_byte	b(8)
i = 2	
}	
do {	
itu_t_t35_payload_byte	b(8)
i++	
} while(i < payloadSize)	
}	

D.2.7 User data unregistered SEI message syntax

user_data_unregistered(payloadSize) {	**Descriptor**
uuid_iso_iec_11578	u(128)
for(i = 16; i < payloadSize; i++)	
user_data_payload_byte	b(8)
}	

D.2.8 Recovery point SEI message syntax

recovery_point(payloadSize) {	**Descriptor**
recovery_poc_cnt	se(v)
exact_match_flag	u(1)
broken_link_flag	u(1)
}	

D.2.9 Scene information SEI message syntax

scene_info(payloadSize) {	**Descriptor**
scene_info_present_flag	u(1)
if(scene_info_present_flag) {	
prev_scene_id_valid_flag	u(1)
scene_id	ue(v)
scene_transition_type	ue(v)
if(scene_transition_type > 3)	
second_scene_id	ue(v)
}	
}	

D.2.10 Picture snapshot SEI message syntax

picture_snapshot(payloadSize) {	**Descriptor**
snapshot_id	ue(v)
}	

D.2.11 Progressive refinement segment start SEI message syntax

progressive_refinement_segment_start(payloadSize) {	**Descriptor**
progressive_refinement_id	ue(v)
pic_order_cnt_delta	ue(v)
}	

D.2.12 Progressive refinement segment end SEI message syntax

progressive_refinement_segment_end(payloadSize) {	**Descriptor**
progressive_refinement_id	ue(v)
}	

D.2.13 Film grain characteristics SEI message syntax

film_grain_characteristics(payloadSize) {	**Descriptor**
film_grain_characteristics_cancel_flag	u(1)
if(!film_grain_characteristics_cancel_flag) {	
film_grain_model_id	u(2)
separate_colour_description_present_flag	u(1)
if(separate_colour_description_present_flag) {	
film_grain_bit_depth_luma_minus8	u(3)
film_grain_bit_depth_chroma_minus8	u(3)
film_grain_full_range_flag	u(1)
film_grain_colour_primaries	u(8)
film_grain_transfer_characteristics	u(8)
film_grain_matrix_coeffs	u(8)
}	
blending_mode_id	u(2)
log2_scale_factor	u(4)
for(c = 0; c < 3; c++)	
comp_model_present_flag[c]	u(1)
for(c = 0; c < 3; c++)	
if(comp_model_present_flag[c]) {	
num_intensity_intervals_minus1[c]	u(8)
num_model_values_minus1[c]	u(3)
for(i = 0; i <= num_intensity_intervals_minus1[c]; i++) {	
intensity_interval_lower_bound[c][i]	u(8)
intensity_interval_upper_bound[c][i]	u(8)
for(j = 0; j <= num_model_values_minus1[c]; j++)	
comp_model_value[c][i][j]	se(v)
}	
}	
film_grain_characteristics_persistence_flag	u(1)
}	
}	

D.2.14 Post-filter hint SEI message syntax

post_filter_hint(payloadSize) {	**Descriptor**
filter_hint_size_y	ue(v)
filter_hint_size_x	ue(v)
filter_hint_type	u(2)
for(cIdx = 0; cIdx < (chroma_format_idc = = 0 ? 1 : 3); cIdx++)	
for(cy = 0; cy < filter_hint_size_y; cy++)	
for(cx = 0; cx < filter_hint_size_x; cx++)	
filter_hint_value[cIdx][cy][cx]	se(v)
}	

D.2.15 Tone mapping information SEI message syntax

tone_mapping_info(payloadSize) {	**Descriptor**
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
tone_map_persistence_flag	u(1)
coded_data_bit_depth	u(8)
target_bit_depth	u(8)
tone_map_model_id	ue(v)
if(tone_map_model_id = = 0) {	
min_value	u(32)
max_value	u(32)
} else if(tone_map_model_id = = 1) {	
sigmoid_midpoint	u(32)
sigmoid_width	u(32)
} else if(tone_map_model_id = = 2)	
for(i = 0; i < (1 << target_bit_depth); i++)	
start_of_coded_interval[i]	u(v)
else if(tone_map_model_id = = 3) {	
num_pivots	u(16)
for(i = 0; i < num_pivots; i++) {	
coded_pivot_value[i]	u(v)
target_pivot_value[i]	u(v)
}	
} else if(tone_map_model_id = = 4) {	
camera_iso_speed_idc	u(8)
if(camera_iso_speed_idc = = EXTENDED_ISO)	
camera_iso_speed_value	u(32)
exposure_idx_idc	u(8)
if(exposure_idx_idc = = EXTENDED_ISO)	
exposure_idx_value	u(32)
exposure_compensation_value_sign_flag	u(1)
exposure_compensation_value_numerator	u(16)
exposure_compensation_value_denom_idc	u(16)
ref_screen_luminance_white	u(32)
extended_range_white_level	u(32)
nominal_black_level_code_value	u(16)
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
}	
}	
}	

D.2.16 Frame packing arrangement SEI message syntax

frame_packing_arrangement(payloadSize) {	**Descriptor**
frame_packing_arrangement_id	ue(v)
frame_packing_arrangement_cancel_flag	u(1)
if(!frame_packing_arrangement_cancel_flag) {	
frame_packing_arrangement_type	u(7)
quincunx_sampling_flag	u(1)
content_interpretation_type	u(6)
spatial_flipping_flag	u(1)
frame0_flipped_flag	u(1)
field_views_flag	u(1)
current_frame_is_frame0_flag	u(1)
frame0_self_contained_flag	u(1)
frame1_self_contained_flag	u(1)
if(!quincunx_sampling_flag && frame_packing_arrangement_type != 5) {	
frame0_grid_position_x	u(4)
frame0_grid_position_y	u(4)
frame1_grid_position_x	u(4)
frame1_grid_position_y	u(4)
}	
frame_packing_arrangement_reserved_byte	u(8)
frame_packing_arrangement_persistence_flag	u(1)
}	
upsampled_aspect_ratio_flag	u(1)
}	

D.2.17 Display orientation SEI message syntax

display_orientation(payloadSize) {	**Descriptor**
display_orientation_cancel_flag	u(1)
if(!display_orientation_cancel_flag) {	
hor_flip	u(1)
ver_flip	u(1)
anticlockwise_rotation	u(16)
display_orientation_persistence_flag	u(1)
}	
}	

D.2.18 Green metadata SEI message syntax

The syntax for green metadata SEI message is specified in ISO/IEC 23001-11 (Green metadata). Green metadata facilitates reduced power consumption in decoders, encoders, displays and in media selection.

D.2.19 Structure of pictures information SEI message syntax

structure_of_pictures_info(payloadSize) {	**Descriptor**
sop_seq_parameter_set_id	ue(v)
num_entries_in_sop_minus1	ue(v)
for(i = 0; i <= num_entries_in_sop_minus1; i++) {	
sop_vcl_nut[i]	u(6)
sop_temporal_id[i]	u(3)
if(sop_vcl_nut[i] != IDR_W_RADL && sop_vcl_nut[i] != IDR_N_LP)	
sop_short_term_rps_idx[i]	ue(v)
if(i > 0)	
sop_poc_delta[i]	se(v)
}	
}	

D.2.20 Decoded picture hash SEI message syntax

decoded_picture_hash(payloadSize) {	**Descriptor**
hash_type	u(8)
for(cIdx = 0; cIdx < (chroma_format_idc = = 0 ? 1 : 3); cIdx++)	
if(hash_type = = 0)	
for(i = 0; i < 16; i++)	
picture_md5[cIdx][i]	b(8)
else if(hash_type = = 1)	
picture_crc[cIdx]	u(16)
else if(hash_type = = 2)	
picture_checksum[cIdx]	u(32)
}	

D.2.21 Active parameter sets SEI message syntax

active_parameter_sets(payloadSize) {	**Descriptor**
active_video_parameter_set_id	u(4)
self_contained_cvs_flag	u(1)
no_parameter_set_update_flag	u(1)
num_sps_ids_minus1	ue(v)
for(i = 0; i <= num_sps_ids_minus1; i++)	
active_seq_parameter_set_id[i]	ue(v)
for(i = vps_base_layer_internal_flag; i <= MaxLayersMinus1; i++)	
layer_sps_idx[i]	ue(v)
}	

D.2.22 Decoding unit information SEI message syntax

decoding_unit_info(payloadSize) {	**Descriptor**
decoding_unit_idx	ue(v)
if(!sub_pic_cpb_params_in_pic_timing_sei_flag)	
du_spt_cpb_removal_delay_increment	u(v)
dpb_output_du_delay_present_flag	u(1)
if(dpb_output_du_delay_present_flag)	
pic_spt_dpb_output_du_delay	u(v)
}	

D.2.23 Temporal sub-layer zero index SEI message syntax

temporal_sub_layer_zero_idx(payloadSize) {	**Descriptor**
temporal_sub_layer_zero_idx	u(8)
irap_pic_id	u(8)
}	

D.2.24 Scalable nesting SEI message syntax

scalable_nesting(payloadSize) {	**Descriptor**
bitstream_subset_flag	u(1)
nesting_op_flag	u(1)
if(nesting_op_flag) {	
default_op_flag	u(1)
nesting_num_ops_minus1	ue(v)
for(i = default_op_flag; i <= nesting_num_ops_minus1; i++) {	
nesting_max_temporal_id_plus1[i]	u(3)
nesting_op_idx[i]	ue(v)
}	
} else {	
all_layers_flag	u(1)
if(!all_layers_flag) {	
nesting_no_op_max_temporal_id_plus1	u(3)
nesting_num_layers_minus1	ue(v)
for(i = 0; i <= nesting_num_layers_minus1; i++)	
nesting_layer_id[i]	u(6)
}	
}	
while(!byte_aligned( ))	
nesting_zero_bit /* equal to 0 */	u(1)
do	
sei_message( )	
while(more_rbsp_data( ))	
}	

D.2.25 Region refresh information SEI message syntax

region_refresh_info(payloadSize) {	**Descriptor**
refreshed_region_flag	u(1)
}	

D.2.26 No display SEI message syntax

no_display(payloadSize) {	**Descriptor**
}	

D.2.27 Time code SEI message syntax

time_code(payloadSize) {	**Descriptor**
num_clock_ts	u(2)
for(i = 0; i < num_clock_ts; i++) {	
clock_timestamp_flag[i]	u(1)
if(clock_timestamp_flag[i]) {	
units_field_based_flag[i]	u(1)
counting_type[i]	u(5)
full_timestamp_flag[i]	u(1)
discontinuity_flag[i]	u(1)
cnt_dropped_flag[i]	u(1)
n_frames[i]	u(9)
if(full_timestamp_flag[i]) {	
seconds_value[i] /* 0..59 */	u(6)
minutes_value[i] /* 0..59 */	u(6)
hours_value[i] /* 0..23 */	u(5)
} else {	
seconds_flag[i]	u(1)
if(seconds_flag[i]) {	
seconds_value[i] /* 0..59 */	u(6)
minutes_flag[i]	u(1)
if(minutes_flag[i]) {	
minutes_value[i] /* 0..59 */	u(6)
hours_flag[i]	u(1)
if(hours_flag[i])	
hours_value[i] /* 0..23 */	u(5)
}	
}	
}	
time_offset_length[i]	u(5)
if(time_offset_length[i] > 0)	
time_offset_value[i]	i(v)
}	
}	
}	

D.2.28 Mastering display colour volume SEI message syntax

mastering_display_colour_volume(payloadSize) {	**Descriptor**
for(c = 0; c < 3; c++) {	
display_primaries_x[c]	u(16)
display_primaries_y[c]	u(16)
}	
white_point_x	u(16)
white_point_y	u(16)
max_display_mastering_luminance	u(32)
min_display_mastering_luminance	u(32)
}	

D.2.29 Segmented rectangular frame packing arrangement SEI message syntax

segmented_rect_frame_packing_arrangement(payloadSize) {	**Descriptor**
segmented_rect_frame_packing_arrangement_cancel_flag	u(1)
if(!segmented_rect_frame_packing_arrangement_cancel_flag) {	
segmented_rect_content_interpretation_type	u(2)
segmented_rect_frame_packing_arrangement_persistence_flag	u(1)
}	
}	

D.2.30 Temporal motion-constrained tile sets SEI message syntax

temporal_motion_constrained_tile_sets(payloadSize) {	**Descriptor**
mc_all_tiles_exact_sample_value_match_flag	u(1)
each_tile_one_tile_set_flag	u(1)
if(!each_tile_one_tile_set_flag) {	
limited_tile_set_display_flag	u(1)
num_sets_in_message_minus1	ue(v)
for(i = 0; i <= num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue(v)
if(limited_tile_set_display_flag)	
display_tile_set_flag[i]	u(1)
num_tile_rects_in_set_minus1[i]	ue(v)
for(j = 0; j <= num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_idx[i][j]	ue(v)
bottom_right_tile_idx[i][j]	ue(v)
}	
if(!mc_all_tiles_exact_sample_value_match_flag)	
mc_exact_sample_value_match_flag[i]	u(1)
mcts_tier_level_idc_present_flag[i]	u(1)
if(mcts_tier_level_idc_present_flag[i]) {	
mcts_tier_flag[i]	u(1)
mcts_level_idc[i]	u(8)
}	
}	
} else {	
max_mcs_tier_level_idc_present_flag	u(1)
if(mcts_max_tier_level_idc_present_flag) {	
mcts_max_tier_flag	u(1)
mcts_max_level_idc	u(8)
}	
}	
}	

D.2.31 Chroma resampling filter hint SEI message syntax

chroma_resampling_filter_hint(payloadSize) {	**Descriptor**
ver_chroma_filter_idc	u(8)
hor_chroma_filter_idc	u(8)
ver_filtering_field_processing_flag	u(1)
if(ver_chroma_filter_idc = = 1 || hor_chroma_filter_idc = = 1) {	
target_format_idc	ue(v)
if(ver_chroma_filter_idc = = 1) {	
num_vertical_filters	ue(v)
for(i = 0; i < num_vertical_filters; i++) {	
ver_tap_length_minus1[i]	ue(v)
for(j = 0; j <= ver_tap_length_minus1[i]; j++)	
ver_filter_coeff[i][j]	se(v)
}	
}	
if(hor_chroma_filter_idc = = 1) {	
num_horizontal_filters	ue(v)
for(i = 0; i < num_horizontal_filters; i++) {	
hor_tap_length_minus1[i]	ue(v)
for(j = 0; j <= hor_tap_length_minus1[i]; j++)	
hor_filter_coeff[i][j]	se(v)
}	
}	
}	
}	

D.2.32 Knee function information SEI message syntax

knee_function_info(payloadSize) {	**Descriptor**
knee_function_id	ue(v)
knee_function_cancel_flag	u(1)
if(!knee_function_cancel_flag) {	
knee_function_persistence_flag	u(1)
input_d_range	u(32)
input_disp_luminance	u(32)
output_d_range	u(32)
output_disp_luminance	u(32)
num_knee_points_minus1	ue(v)
for(i = 0; i <= num_knee_points_minus1; i++) {	
input_knee_point[i]	u(10)
output_knee_point[i]	u(10)
}	
}	
}	

D.2.33 Colour remapping information SEI message syntax

colour_remapping_info(payloadSize) {	**Descriptor**
colour_remap_id	ue(v)
colour_remap_cancel_flag	u(1)
if(!colour_remap_cancel_flag) {	
colour_remap_persistence_flag	u(1)
colour_remap_video_signal_info_present_flag	u(1)
if(colour_remap_video_signal_info_present_flag) {	
colour_remap_full_range_flag	u(1)
colour_remap_primaries	u(8)
colour_remap_transfer_function	u(8)
colour_remap_matrix_coefficients	u(8)
}	
colour_remap_input_bit_depth	u(8)
colour_remap_output_bit_depth	u(8)
for(c = 0; c < 3; c++) {	
pre_lut_num_val_minus1[c]	u(8)
if(pre_lut_num_val_minus1[c] > 0)	
for(i = 0; i <= pre_lut_num_val_minus1[c]; i++) {	
pre_lut_coded_value[c][i]	u(v)
pre_lut_target_value[c][i]	u(v)
}	
}	
colour_remap_matrix_present_flag	u(1)
if(colour_remap_matrix_present_flag) {	
log2_matrix_denom	u(4)
for(c = 0; c < 3; c++)	
for(i = 0; i < 3; i++)	
colour_remap_coeffs[c][i]	se(v)
}	
for(c = 0; c < 3; c++) {	
post_lut_num_val_minus1[c]	u(8)
if(post_lut_num_val_minus1[c] > 0)	
for(i = 0; i <= post_lut_num_val_minus1[c]; i++) {	
post_lut_coded_value[c][i]	u(v)
post_lut_target_value[c][i]	u(v)
}	
}	
}	
}	

D.2.34 Deinterlaced field identification SEI message syntax

deinterlaced_field_indentification(payloadSize) {	**Descriptor**
deinterlaced_picture_source_parity_flag	u(1)
}	

D.2.35 Content light level information SEI message syntax

content_light_level_info(payloadSize) {	**Descriptor**
max_content_light_level	u(16)
max_pic_average_light_level	u(16)
}	

D.2.36 Dependent random access point indication SEI message syntax

dependent_rap_indication(payloadSize) {	**Descriptor**
}	

D.2.37 Coded region completion SEI message syntax

coded_region_completion(payloadSize) {	**Descriptor**
next_segment_address	ue(v)
if(next_segment_address > 0)	
independent_slice_segment_flag	u(1)
}	

D.2.38 Alternative transfer characteristics information SEI message syntax

alternative_transfer_characteristics (payloadSize) {	**Descriptor**
preferred_transfer_characteristics	u(8)
}	

D.2.39 Ambient viewing environment SEI message syntax

ambient_viewing_environment(payloadSize) {	**Descriptor**
ambient_illuminance	u(32)
ambient_light_x	u(16)
ambient_light_y	u(16)
}	

D.2.40 Content colour volume SEI message syntax

content_colour_volume(payloadSize) {	**Descriptor**
ccv_cancel_flag	u(1)
if(!ccv_cancel_flag) {	
ccv_persistence_flag	u(1)
ccv_primaries_present_flag	u(1)
ccv_min_luminance_value_present_flag	u(1)
ccv_max_luminance_value_present_flag	u(1)
ccv_avg_luminance_value_present_flag	u(1)
ccv_reserved_zero_2bits	u(2)
if(ccv_primaries_present_flag)	
for(c = 0; c < 3; c++) {	
ccv_primaries_x[c]	i(32)
ccv_primaries_y[c]	i(32)
}	
if(ccv_min_luminance_value_present_flag)	
ccv_min_luminance_value	u(32)
if(ccv_max_luminance_value_present_flag)	
ccv_max_luminance_value	u(32)
if(ccv_avg_luminance_value_present_flag)	
ccv_avg_luminance_value	u(32)
}	
}	

D.2.41 Syntax of omnidirectional video specific SEI messages

D.2.41.1 Equirectangular projection SEI message syntax

equirectangular_projection(payloadSize) {	**Descriptor**
erp_cancel_flag	u(1)
if(!erp_cancel_flag) {	
erp_persistence_flag	u(1)
erp_guard_band_flag	u(1)
erp_reserved_zero_2bits	u(2)
if(erp_guard_band_flag = = 1) {	
erp_guard_band_type	u(3)
erp_left_guard_band_width	u(8)
erp_right_guard_band_width	u(8)
}	
}	
}	

D.2.41.2 Cubemap projection SEI message syntax

cubemap_projection(payloadSize) {	**Descriptor**
cmp_cancel_flag	u(1)
if(!cmp_cancel_flag)	
cmp_persistence_flag	u(1)
}	

D.2.41.3 Fisheye video information SEI message syntax

fisheye_video_info(payloadSize) {	**Descriptor**
fisheye_cancel_flag	u(1)
if(!fisheye_cancel_flag) {	
fisheye_persistence_flag	u(1)
fisheye_view_dimension_idc	u(3)
fisheye_reserved_zero_3bits	u(3)
fisheye_num_active_areas_minus1	u(8)
for(i = 0; i <= fisheye_num_active_areas_minus1; i++) {	
fisheye_circular_region_centre_x[i]	u(32)
fisheye_circular_region_centre_y[i]	u(32)
fisheye_rect_region_top[i]	u(32)
fisheye_rect_region_left[i]	u(32)
fisheye_rect_region_width[i]	u(32)
fisheye_rect_region_height[i]	u(32)
fisheye_circular_region_radius[i]	u(32)
fisheye_scene_radius[i]	u(32)
fisheye_camera_centre_azimuth[i]	i(32)
fisheye_camera_centre_elevation[i]	i(32)
fisheye_camera_centre_tilt[i]	i(32)
fisheye_camera_centre_offset_x[i]	u(32)
fisheye_camera_centre_offset_y[i]	u(32)
fisheye_camera_centre_offset_z[i]	u(32)
fisheye_field_of_view[i]	u(32)
fisheye_num_polynomial_coeffs[i]	u(16)
for(j = 0; j < fisheye_num_polynomial_coeffs[i]; j++)	
fisheye_polynomial_coeff[i][j]	i(32)
}	
}	
}	

D.2.41.4 Sphere rotation SEI message syntax

sphere_rotation(payloadSize) {	**Descriptor**
sphere_rotation_cancel_flag	u(1)
if(!sphere_rotation_cancel_flag) {	
sphere_rotation_persistence_flag	u(1)
sphere_rotation_reserved_zero_6bits	u(6)
yaw_rotation	i(32)
pitch_rotation	i(32)
roll_rotation	i(32)
}	
}	

D.2.41.5 Region-wise packing SEI message syntax

regionwise_packing(payloadSize) {	**Descriptor**
rwp_cancel_flag	u(1)
if(!rwp_cancel_flag) {	
rwp_persistence_flag	u(1)
constituent_picture_matching_flag	u(1)
rwp_reserved_zero_5bits	u(5)
num_packed_regions	u(8)
proj_picture_width	u(32)
proj_picture_height	u(32)
packed_picture_width	u(16)
packed_picture_height	u(16)
for(i = 0; i < num_packed_regions; i++) {	
rwp_reserved_zero_4bits[i]	u(4)
rwp_transform_type[i]	u(3)
rwp_guard_band_flag[i]	u(1)
proj_region_width[i]	u(32)
proj_region_height[i]	u(32)
proj_region_top[i]	u(32)
proj_region_left[i]	u(32)
packed_region_width[i]	u(16)
packed_region_height[i]	u(16)
packed_region_top[i]	u(16)
packed_region_left[i]	u(16)
if(rwp_guard_band_flag[i]) {	
rwp_left_guard_band_width[i]	u(8)
rwp_right_guard_band_width[i]	u(8)
rwp_top_guard_band_height[i]	u(8)
rwp_bottom_guard_band_height[i]	u(8)
rwp_guard_band_not_used_for_pred_flag[i]	u(1)
for(j = 0; j < 4; j++)	
rwp_guard_band_type[i][j]	u(3)
rwp_guard_band_reserved_zero_3bits[i]	u(3)
}	
}	
}	
}	

D.2.41.6 Omnidirectional viewport SEI message syntax

omni_viewport(payloadSize) {	**Descriptor**
omni_viewport_id	u(10)
omni_viewport_cancel_flag	u(1)
if(!omni_viewport_cancel_flag) {	
omni_viewport_persistence_flag	u(1)
omni_viewport_cnt_minus1	u(4)
for(i = 0; i <= omni_viewport_cnt_minus1; i++) {	
omni_viewport_azimuth_centre[i]	i(32)
omni_viewport_elevation_centre[i]	i(32)
omni_viewport_tilt_centre[i]	i(32)
omni_viewport_hor_range[i]	u(32)
omni_viewport_ver_range[i]	u(32)
}	
}	
}	

D.2.42 Regional nesting SEI message syntax

regional_nesting(payloadSize) {	**Descriptor**
regional_nesting_id	u(16)
regional_nesting_num_rect_regions	u(8)
for(i = 0; i < regional_nesting_num_rect_regions; i++) {	
regional_nesting_rect_region_id[i]	u(8)
regional_nesting_rect_left_offset[i]	u(16)
regional_nesting_rect_right_offset[i]	u(16)
regional_nesting_rect_top_offset[i]	u(16)
regional_nesting_rect_bottom_offset[i]	u(16)
}	
num_sei_messages_in_regional_nesting_minus1	u(8)
for(i = 0; i <= num_sei_messages_in_regional_nesting_minus1; i++) {	
num_regions_for_sei_message[i]	u(8)
for(j = 0; j < num_regions_for_sei_message[i]; j++)	
regional_nesting_sei_region_idx[i][j]	u(8)
sei_message( )	
}	
}	

D.2.43 Motion-constrained tile sets extraction information sets SEI message syntax

mcts_extraction_info_sets(payloadSize) {	**Descriptor**
num_info_sets_minus1	ue(v)
for(i = 0; i <= num_info_sets_minus1; i++) {	
num_mcts_sets_minus1[i]	ue(v)
for(j = 0; j <= num_mcts_sets_minus1[i]; j++) {	
num_mcts_in_set_minus1[i][j]	ue(v)
for(k = 0; k <= num_mcts_in_set_minus1[i][j]; k++)	
idx_of_mcts_in_set[i][j][k]	ue(v)
}	
slice_reordering_enabled_flag[i]	u(1)
if(slice_reordering_enabled_flag[i]) {	
num_slice_segments_minus1[i]	ue(v)
for(j = 0; j <= num_slice_segments_minus1[i]; j++)	
output_slice_segment_address[i][j]	u(v)
}	
num_vps_in_info_set_minus1[i]	ue(v)
for(j = 0; j <= num_vps_in_info_set_minus1[i]; j++)	
vps_rbsp_data_length[i][j]	ue(v)
num_sps_in_info_set_minus1[i]	ue(v)
for(j = 0; j <= num_sps_in_info_set_minus1[i]; j++)	
sps_rbsp_data_length[i][j]	ue(v)
num_pps_in_info_set_minus1[i]	ue(v)
for(j = 0; j <= num_pps_in_info_set_minus1[i]; j++) {	
pps_nuh_temporal_id_plus1[i][j]	u(3)
pps_rbsp_data_length[i][j]	ue(v)
}	
while(!byte_aligned( ))	
mcts_alignment_bit_equal_to_zero	f(1)
for(j = 0; j <= num_vps_in_info_set_minus1[i]; j++)	
for(k = 0; k < vps_rbsp_data_length[i][j]; k++)	
vps_rbsp_data_byte[i][j][k]	u(8)
for(j = 0; j <= num_sps_in_info_set_minus1[i]; j++)	
for(k = 0; k < sps_rbsp_data_length[i][j]; k++)	
sps_rbsp_data_byte[i][j][k]	u(8)
for(j = 0; j <= num_pps_in_info_set_minus1[i]; j++)	
for(k = 0; k < pps_rbsp_data_length[i][j]; k++)	
pps_rbsp_data_byte[i][j][k]	u(8)
}	
}	

D.2.44 Motion-constrained tile sets extraction information nesting SEI message syntax

mcts_extraction_info_nesting(payloadSize) {	**Descriptor**
all_mcts_flag	u(1)
if(!all_mcts_flag) {	
num_associated_mcts_minus1	ue(v)
for(i = 0; i <= num_associated_mcts_minus1; i++)	
idx_of_associated_mcts[i]	ue(v)
}	
num_sei_messages_in_mcts_extraction_nesting_minus1	ue(v)
while(!byte_aligned( ))	
mcts_nesting_zero_bit /* equal to 0 */	u(1)
for(i = 0; i <= num_sei_messages_in_mcts_extraction_nesting_minus1; i++)	
sei_message( )	
}	

D.2.45 SEI manifest SEI message syntax

sei_manifest(payloadSize) {	**Descriptor**
manifest_num_sei_msg_types	u(16)
for(i = 0; i < manifest_num_sei_msg_types; i++) {	
manifest_sei_payload_type[i]	u(16)
manifest_sei_description[i]	u(8)
}	
}	

D.2.46 SEI prefix indication SEI message syntax

sei_prefix_indication(payloadSize) {	**Descriptor**
prefix_sei_payload_type	u(16)
num_sei_prefix_indications_minus1	u(8)
for(i = 0; i <= num_sei_prefix_indications_minus1; i++) {	
num_bits_in_prefix_indication_minus1[i]	u(16)
for(j = 0; j <= num_bits_in_prefix_indication_minus1[i]; j++)	
sei_prefix_data_bit[i][j]	u(1)
while(!byte_aligned( ))	
byte_alignment_bit_equal_to_one /* equal to 1 */	f(1)
}	
}	

D.2.47 Annotated regions SEI message syntax

annotated_regions(payloadSize) {	**Descriptor**
ar_cancel_flag	u(1)
if(!ar_cancel_flag) {	
ar_not_optimized_for_viewing_flag	u(1)
ar_true_motion_flag	u(1)
ar_occluded_object_flag	u(1)
ar_partial_object_flag_present_flag	u(1)
ar_object_label_present_flag	u(1)
ar_object_confidence_info_present_flag	u(1)
if(ar_object_confidence_info_present_flag)	
ar_object_confidence_length_minus1	u(4)
if(ar_object_label_present_flag) {	
ar_object_label_language_present_flag	u(1)
if(ar_object_label_language_present_flag) {	
while(!byte_aligned( ))	
ar_bit_equal_to_zero /* equal to 0 */	f(1)
ar_object_label_language	st(v)
}	
ar_num_label_updates	ue(v)
for(i = 0; i < ar_num_label_updates; i++) {	
ar_label_idx[i]	ue(v)
ar_label_cancel_flag	u(1)
LabelAssigned[ar_label_idx[i]] = !ar_label_cancel_flag	
if(!ar_label_cancel_flag) {	
while(!byte_aligned( ))	
ar_bit_equal_to_zero /* equal to 0 */	f(1)
ar_label[ar_label_idx[i]]	st(v)
}	
}	
}	
ar_num_object_updates	ue(v)
for(i = 0; i < ar_num_object_updates; i++) {	
ar_object_idx[i]	ue(v)
ar_object_cancel_flag	u(1)
ObjectTracked[ar_object_idx[i]] = !ar_object_cancel_flag	
if(!ar_object_cancel_flag) {	
if(ar_object_label_present_flag) {	
ar_object_label_update_flag	u(1)
if(ar_object_label_update_flag)	
ar_object_label_idx[ar_object_idx[i]]	ue(v)
}	
ar_bounding_box_update_flag	u(1)
if(ar_bounding_box_update_flag) {	
ar_bounding_box_cancel_flag	u(1)
ObjectBoundingBoxAvail[ar_object_idx[i]] = !ar_bounding_box_cancel_flag	

if(!ar_bounding_box_cancel_flag) {	
ar_bounding_box_top[ar_object_idx[i]]	u(16)
ar_bounding_box_left[ar_object_idx[i]]	u(16)
ar_bounding_box_width[ar_object_idx[i]]	u(16)
ar_bounding_box_height[ar_object_idx[i]]	u(16)
if(ar_partial_object_flag_present_flag)	
ar_partial_object_flag[ar_object_idx[i]]	u(1)
if(ar_object_confidence_info_present_flag)	
ar_object_confidence[ar_object_idx[i]]	u(v)
}	
}	
}	
}	
}	
}	

D.2.48 Shutter interval information SEI message syntax

shutter_interval_info(payloadSize) {	**Descriptor**
sii_time_scale	u(32)
fixed_shutter_interval_within_clvs_flag	u(1)
if(fixed_shutter_interval_within_clvs_flag)	
sii_num_units_in_shutter_interval	u(32)
else {	
sii_max_sub_layers_minus1	u(3)
for(i = 0; i <= sii_max_sub_layers_minus1; i++)	
sub_layer_num_units_in_shutter_interval[i]	u(32)
}	
}	

D.2.49 Reserved SEI message syntax

reserved_sei_message(payloadSize) {	**Descriptor**
for(i = 0; i < payloadSize; i++)	
reserved_sei_message_payload_byte	b(8)
}	

D.3 SEI payload semantics

D.3.1 General SEI payload semantics

reserved_payload_extension_data shall not be present in bitstreams conforming to this version of this Specification. However, decoders conforming to this version of this Specification shall ignore the presence and value of reserved_payload_extension_data. When present, the length, in bits, of reserved_payload_extension_data is equal to 8 * payloadSize − nEarlierBits − nPayloadZeroBits − 1, where nEarlierBits is the number of bits in the sei_payload( ) syntax structure that precede the reserved_payload_extension_data syntax element and nPayloadZeroBits is the number of payload_bit_equal_to_zero syntax elements at the end of the sei_payload( ) syntax structure.

payload_bit_equal_to_one shall be equal to 1.

payload_bit_equal_to_zero shall be equal to 0.

NOTE 1 – SEI messages with the same value of payloadType are conceptually the same SEI message regardless of whether they are contained in prefix or suffix SEI NAL units.

NOTE 2 – For SEI messages with payloadType in the range of 0 to 47, inclusive, that are specified in this Specification, the payloadType values are aligned with similar SEI messages specified in Rec. ITU-T H.264 | ISO/IEC 14496-10.

The list SingleLayerSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, and 205.

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, and 205.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, and 205.

NOTE 3 – SingleLayerSeiList consists of the payloadType values of the SEI messages specified in Annex D excluding 0 (buffering period), 1 (picture timing), 4 (user data registered by Recommendation ITU-T T.35), 5 (user data unregistered), 130 (decoding unit information) and 133 (scalable nesting). VclAssociatedSeiList consists of the payloadType values of the SEI messages that, when non-scalable-nested and contained in an SEI NAL unit, infer constraints on the NAL unit header of the SEI NAL unit on the basis of the NAL unit header of the associated VCL NAL unit. PicUnitRepConSeiList consists of the payloadType values of the SEI messages that are subject to the restriction on 8 repetitions per picture unit.

The semantics and persistence scope for each SEI message are specified in the semantics specification for each particular SEI message.

NOTE 4 – Persistence information for SEI messages is informatively summarized in Table D.1.

Table D.1 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Buffering period	The remainder of the bitstream
Picture timing	The access unit containing the SEI message
Pan-scan rectangle	Specified by the syntax of the SEI message
Filler payload	The access unit containing the SEI message
User data registered by Rec. ITU-T T.35	Unspecified
User data unregistered	Unspecified
Recovery point	Specified by the syntax of the SEI message
Scene information	The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains a scene information SEI message or starts a new CLVS
Picture snapshot	The access unit containing the SEI message
Progressive refinement segment start	Specified by the syntax of the SEI message
Progressive refinement segment end	The access unit containing the SEI message
Film grain characteristics	Specified by the syntax of the SEI message
Post-filter hint	The access unit containing the SEI message
Tone mapping information	Specified by the syntax of the SEI message
Frame packing arrangement	Specified by the syntax of the SEI message
Display orientation	Specified by the syntax of the SEI message
Green metadata	Specified by the syntax of the SEI message
Structure of pictures information	The set of pictures in the coded layer-wise video sequence (CLVS) that correspond to entries listed in the SEI message
Decoded picture hash	The access unit containing the SEI message
Active parameter sets	The CVS containing the SEI message
Decoding unit information	The decoding unit containing the SEI message
Temporal sub-layer zero index	The access unit containing the SEI message
Scalable nesting	Depending on the scalable-nested SEI messages. Each scalable-nested SEI message has the same persistence scope as if the SEI message was not scalable-nested

Table D.1 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Region refresh information	The set of VCL NAL units within the access unit starting from the VCL NAL unit following the SEI message up to but not including the VCL NAL unit following the next SEI NAL unit containing a region refresh information SEI message (if any)
No display	The access unit containing the SEI message
Time code	The access unit containing the SEI message
Mastering display colour volume	The CLVS containing the SEI message
Segmented rectangular frame packing arrangement	Specified by the syntax of the SEI message
Temporal motion-constrained tile sets	The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains an SEI message of the same type or starts a new CLVS
Chroma resampling filter hint	The CLVS containing the SEI message
Knee function information	Specified by the syntax of the SEI message
Colour remapping information	Specified by the syntax of the SEI message
Deinterlaced field identification	One or more pictures associated with the access unit containing the SEI message
Content light level information	The CLVS containing the SEI message
Dependent random access point indication	The access unit containing the SEI message
Coded region completion	The current slice segment associated with the SEI message
Alternative transfer characteristics	The CLVS containing the SEI message
Ambient viewing environment	The CLVS containing the SEI message
Content colour volume	Specified by the syntax of the SEI message
Equirectangular projection	Specified by the syntax of the SEI message
Cubemap projection	Specified by the syntax of the SEI message
Fisheye video information	Specified by the syntax of the SEI message
Sphere rotation	Specified by the syntax of the SEI message
Region-wise packing	Specified by the syntax of the SEI message
Omnidirectional viewport	Specified by the syntax of the SEI message
Regional nesting	Depending on the region-nested SEI messages; each region-nested SEI message has the same persistence scope as if the SEI message was non-region-nested
Motion-constrained tile sets extraction information sets	The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains an SEI message of the same type or starts a new CLVS
Motion-constrained tile sets extraction information nesting	The access unit containing the SEI message
SEI manifest	The CVS containing the SEI message
SEI prefix indication	The CVS containing the SEI message
Annotated regions	Specified by the syntax of the SEI message
Shutter interval information	The CLVS containing the SEI message

The values of some SEI message syntax elements, including pan_scan_rect_id, scene_id, second_scene_id, snapshot_id, progressive_refinement_id, tone_map_id, frame_packing_arrangement_id, mcts_id[i], knee_function_id, colour_remap_id, ilcts_id[i], and regional_nesting_id, are split into two sets of value ranges, where the first set is specified as "may be used as determined by the application", and the second set is specified as "reserved for future use by ITU-T |

ISO/IEC". Applications should be cautious of potential "collisions" of the interpretation for values of these syntax elements belonging to the first set of value ranges. Since different applications might use these IDs having values in the first set of value ranges for different purposes, particular care should be exercised in the design of encoders that generate SEI messages with these IDs having values in the first set of value ranges, and in the design of decoders that interpret SEI messages with these IDs. This Specification does not define any management for these values. These IDs having values in the first set of value ranges might only be suitable for use in contexts in which "collisions" of usage (i.e., different definitions of the syntax and semantics of an SEI message with one of these IDs having the same value in the first set of value ranges) are unimportant, or not possible, or are managed – e.g., defined or managed in the controlling application or transport specification, or by controlling the environment in which bitstreams are distributed.

It is a requirement of bitstream conformance that when a prefix SEI message with payloadType equal to 17 (progressive refinement segment end) or 22 (post-filter hint) is present in an access unit, a suffix SEI message with the same value of payloadType shall not be present in the same access unit.

It is a requirement of bitstream conformance that the following restrictions apply on containing of SEI messages in SEI NAL units:

– An SEI NAL unit containing an active parameter sets SEI message shall contain only one active parameter sets SEI message and shall not contain any other SEI messages.

– When an SEI NAL unit contains a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, or a non-scalable-nested decoding unit information SEI message, the SEI NAL unit shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information).

– When an SEI NAL unit contains a scalable-nested buffering period SEI message, a scalable-nested picture timing SEI message, or a scalable-nested decoding unit information SEI message, the SEI NAL unit shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing), 130 (decoding unit information) or 133 (scalable nesting).

Let prevVclNalUnitInAu of an SEI NAL unit or an SEI message be the preceding VCL NAL unit in decoding order, if any, in the same access unit, and nextVclNalUnitInAu of an SEI NAL unit or an SEI message be the next VCL NAL unit in decoding order, if any, in the same access unit.

It is a requirement of bitstream conformance that the following restrictions apply on decoding order of SEI messages:

– When an SEI NAL unit containing an active parameter sets SEI message is present in an access unit, it shall be the first SEI NAL unit that follows the prevVclNalUnitInAu of the SEI NAL unit and precedes the nextVclNalUnitInAu of the SEI NAL unit.

– When a non-scalable-nested buffering period SEI message is present in an access unit, it shall not follow any other SEI message that follows the prevVclNalUnitInAu of the buffering period SEI message and precedes the nextVclNalUnitInAu of the buffering period SEI message, other than an active parameter sets SEI message.

– When a non-scalable-nested picture timing SEI message is present in an access unit, it shall not follow any other SEI message that follows the prevVclNalUnitInAu of the picture timing SEI message and precedes the nextVclNalUnitInAu of the picture timing SEI message, other than an active parameter sets SEI message or a non-scalable-nested buffering period SEI message.

– When a non-scalable-nested decoding unit information SEI message is present in an access unit, it shall not follow any other SEI message in the same access unit that follows the prevVclNalUnitInAu of the decoding unit information SEI message and precedes the nextVclNalUnitInAu of the decoding unit information SEI message, other than an active parameter sets SEI message, a non-scalable-nested buffering period SEI message, or a non-scalable-nested picture timing SEI message.

– When a scalable-nested buffering period SEI message, a scalable-nested picture timing SEI message, or a scalable-nested decoding unit information SEI message is contained in a scalable nesting SEI message in an access unit, the scalable nesting SEI message shall not follow any other SEI message that follows the prevVclNalUnitInAu of the scalable nesting SEI message and precedes the nextVclNalUnitInAu of the scalable nesting SEI message, other than an active parameter sets SEI message, a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, a non-scalable-nested decoding unit information SEI message, or another scalable nesting SEI message that contains a buffering period SEI message, a picture timing SEI message, or a decoding unit information SEI message.

– When payloadType is equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information) for an SEI message, scalable-nested or non-scalable-nested, within the access unit, the SEI NAL unit containing the SEI message shall precede all NAL units of any picture unit that has nuh_layer_id greater than highestAppLayerId, where highestAppLayerId is the greatest value of nuh_layer_id of all the layers in all the operation points that the SEI message applies to.

– When payloadType is equal to any value among VclAssociatedSeiList for an SEI message, scalable-nested or non-scalable-nested, within the access unit, the SEI NAL unit containing the SEI message shall precede all NAL units of any picture unit that has nuh_layer_id greater than highestAppLayerId, where highestAppLayerId is the greatest value of nuh_layer_id of all the layers that the SEI message applies to.

The following applies on the applicable operation points or layers of SEI messages:

– For a non-scalable-nested SEI message, when payloadType is equal to 0 (buffering period) or 130 (decoding unit information), the non-scalable-nested SEI message applies to the operation point that has OpTid equal to the greatest value of nuh_temporal_id_plus1 among all VCL NAL units in the bitstream, has OpLayerIdList containing all values of nuh_layer_id in all VCL units in the bitstream, and has only the base layer as the output layer.

– An SEI message that is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0 and has payloadType is equal to 0 (buffering period), 1 (picture timing), or 130 (decoding unit information) applies as specified in Annex C to the layer set as indicated by the scalable nesting SEI message.

– For a non-scalable-nested SEI message, when payloadType is equal to 1 (picture timing), the frame field information carried in the syntax elements pic_struct, source_scan_type and duplicate_flag, when present, in the non-scalable-nested picture timing SEI message applies to the base layer only, while the picture timing information carried in other syntax elements, when present, in the non-scalable-nested picture timing SEI message applies to the operation point that has OpTid equal to the greatest value of nuh_temporal_id_plus1 among all VCL NAL units in the bitstream, has OpLayerIdList containing all values of nuh_layer_id in all VCL units in the bitstream, and has only the base layer as the output layer.

– For a non-scalable-nested SEI message, when payloadType is equal to any value among VclAssociatedSeiList, the non-scalable-nested SEI message applies to the layer for which the VCL NAL units have nuh_layer_id equal to the nuh_layer_id of the SEI NAL unit containing the SEI message.

– An active parameter sets SEI message, which cannot be scalable-nested, applies to all layers in the bitstream.

It is a requirement of bitstream conformance that the following restrictions apply on the values of nuh_layer_id and TemporalId of SEI NAL units:

– When a non-scalable-nested SEI message has payloadType equal to any value among VclAssociatedSeiList, the SEI NAL unit containing the non-scalable-nested SEI message shall have TemporalId equal to the TemporalId of the access unit containing the SEI NAL unit.

– When a non-scalable-nested SEI message has payloadType equal to 0, 1, 129 or 130, the SEI NAL unit containing the non-scalable-nested SEI message shall have nuh_layer_id equal to 0.

– When a non-scalable-nested SEI message has payloadType equal to any value among VclAssociatedSeiList, the SEI NAL unit containing the non-scalable-nested SEI message shall have nuh_layer_id and nuh_temporal_id_plus1 equal to the values of nuh_layer_id and nuh_temporal_id_plus1, respectively, of the VCL NAL unit associated with the SEI NAL unit.

NOTE 4 – For an SEI NAL unit containing a scalable nesting SEI message, the values of TemporalId and nuh_layer_id should be set equal to the lowest value of TemporalId and nuh_layer_id, respectively, of all the sub-layers or operation points the scalable-nested SEI messages apply to unless specified otherwise.

It is a requirement of bitstream conformance that the following restrictions apply on the presence of SEI messages between two VCL NAL units of a picture:

– When there is a prefix SEI message that has payloadType equal to any value among SingleLayerSeiList not equal to 134 (the region refresh information SEI message) or 146 (the coded region completion SEI message), and applies to a picture of a layer layerA present between two VCL NAL units of the picture in decoding order, there shall be a prefix SEI message that is of the same type and applies to the layer layerA in the same access unit preceding the first VCL NAL unit of the picture.

– When there is a suffix SEI message that has payloadType equal to 3 (filler payload), 17 (progressive refinement segment end), 22 (post filter hint) or 132 (decoded picture hash) and applies to a picture of a layer layerA present between two VCL NAL units of the picture in decoding order, there shall be a suffix SEI message that is of the same type and applies to the layer layerA present in the same access unit succeeding the last VCL NAL unit of the picture.

It is a requirement of bitstream conformance that the following restrictions apply on repetition of SEI messages:

– For each of the payloadType values included in PicUnitRepConSeiList, there shall be less than or equal to 8 identical sei_payload( ) syntax structures within a picture unit.

– There shall be less than or equal to 8 identical sei_payload( ) syntax structures with payloadType equal to 130 within a decoding unit.

– The number of identical sei_payload( ) syntax structures with payloadType equal to 134 in a picture unit shall be less than or equal to the number of slice segments in the picture unit.

In the following subclauses of this annex, the following applies:

– The current SEI message refers to the particular SEI message.

– The current access unit refers to the access unit containing the current SEI message.

In the following subclauses of this annex, when a particular SEI message applies to a set of one or more layers (instead of a set of operation points), i.e., when the payloadType value is not equal to one of 0 (buffering period), 1 (picture timing) and 130 (decoding unit information), the following applies:

– The semantics apply independently to each particular layer with nuh_layer_id equal to targetLayerId of the layers to which the particular SEI message applies.

– The current layer refers to the layer with nuh_layer_id equal to targetLayerId.

– The current picture or the current decoded picture refers to the picture with nuh_layer_id equal to targetLayerId (i.e., in the current layer) in the current access unit.

In the following subclauses of this annex, when a particular SEI message applies to a set of one or more operation points (instead of a set of one or more layers), i.e., when the payloadType value is equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information), the following applies:

– When the particular SEI message applies to an operation point that does not include the base layer (i.e., when the SEI message is contained in an SEI NAL unit with nuh_layer_id greater than 0), decoders conforming to a profile specified in Annex A and not supporting the INBLD capability specified in Annex F shall ignore that particular SEI message.

– The semantics apply independently to each particular operation point of the set of operation points to which the particular SEI message applies.

– The current operation point refers to the particular operation point.

– The terms "access unit" and "CVS" apply to the bitstream BitstreamToDecode that is the sub-bitstream of the particular operation point.

D.3.2 Buffering period SEI message semantics

A buffering period SEI message provides initial CPB removal delay and initial CPB removal delay offset information for initialization of the HRD at the position of the associated access unit in decoding order.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0 and the current access unit is a CRA or BLA acces unit, let skippedPictureList be the list of skipped leading pictures consisting of the RASL pictures associated with the CRA or BLA access unit with which the buffering period SEI message is associated.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, a picture is said to be a notDiscardablePic picture when the picture has TemporalId equal to 0 and is not a RASL, RADL or SLNR picture.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, the following applies for the buffering period SEI message syntax and semantics:

– The syntax elements initial_cpb_removal_delay_length_minus1, au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1, and sub_pic_hrd_params_present_flag and the variables NalHrdBpPresentFlag and VclHrdBpPresentFlag are found in or derived from syntax elements found in the hrd_parameters( ) syntax structure that is applicable to at least one of the operation points to which the buffering period SEI message applies.

– The variables CpbSize[i], BitRate[i] and CpbCnt are derived from syntax elements found in the sub_layer_hrd_parameters( ) syntax structure that is applicable to at least one of the operation points to which the buffering period SEI message applies.

– Any two operation points that the buffering period SEI message applies to having different OpTid values tIdA and tIdB indicate that the values of cpb_cnt_minus1[tIdA] and cpb_cnt_minus1[tIdB] coded in the hrd_parameters( ) syntax structure(s) applicable to the two operation points are identical.

– Any two operation points that the buffering period SEI message applies to having different OpLayerIdList values layerIdListA and layerIdListB indicate that the values of nal_hrd_parameters_present_flag and vcl_hrd_parameters_present_flag, respectively, for the two hrd_parameters( ) syntax structures applicable to the two operation points are identical.

– The bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with any of the operation points to which the buffering period SEI message applies.

The presence of buffering period SEI messages for an operation point including the base layer is specified as follows:

– If NalHrdBpPresentFlag is equal to 1 or VclHrdBpPresentFlag is equal to 1, the following applies for each access unit in the CVS:
  – If the access unit is an IRAP access unit, a buffering period SEI message applicable to the operation point shall be associated with the access unit.
  – Otherwise, if the access unit contains a notDiscardablePic, a buffering period SEI message applicable to the operation point may or may not be associated with the access unit.
  – Otherwise, the access unit shall not be associated with a buffering period SEI message applicable to the operation point.
– Otherwise (NalHrdBpPresentFlag and VclHrdBpPresentFlag are both equal to 0), no access unit in the CVS shall be associated with a buffering period SEI message applicable to the operation point.

NOTE 1 – For some applications, frequent presence of buffering period SEI messages may be desirable (e.g., for random access at an IRAP picture or a non-IRAP picture or for bitstream splicing).

bp_seq_parameter_set_id indicates and shall be equal to the sps_seq_parameter_set_id for the SPS that is active for the coded picture associated with the buffering period SEI message. The value of bp_seq_parameter_set_id shall be equal to the value of pps_seq_parameter_set_id in the PPS referenced by the slice_pic_parameter_set_id of the slice segment headers of the coded picture associated with the buffering period SEI message. The value of bp_seq_parameter_set_id shall be in the range of 0 to 15, inclusive.

irap_cpb_params_present_flag equal to 1 specifies the presence of the nal_initial_alt_cpb_removal_delay[i] and nal_initial_alt_cpb_removal_offset[i] or vcl_initial_alt_cpb_removal_delay[i] and vcl_initial_alt_cpb_removal_offset[i] syntax elements. When not present, the value of irap_cpb_params_present_flag is inferred to be equal to 0. When the associated picture is neither a CRA picture nor a BLA picture, the value of irap_cpb_params_present_flag shall be equal to 0.

NOTE 2 – The values of sub_pic_hrd_params_present_flag and irap_cpb_params_present_flag cannot be both equal to 1.

cpb_delay_offset specifies an offset to be used in the derivation of the nominal CPB removal times of access units following, in decoding order, the CRA or BLA access unit associated with the buffering period SEI message when no picture in skippedPictureList is present. The syntax element has a length in bits given by au_cpb_removal_delay_length_minus1 + 1. When not present, the value of cpb_delay_offset is inferred to be equal to 0.

dpb_delay_offset specifies an offset to be used in the derivation of the DPB output times of the CRA or BLA access unit associated with the buffering period SEI message when no picture in skippedPictureList is present. The syntax element has a length in bits given by dpb_output_delay_length_minus1 + 1. When not present, the value of dpb_delay_offset is inferred to be equal to 0.

When the current picture is not the first picture in the bitstream in decoding order, let prevNonDiscardablePic be the preceding picture in decoding order with TemporalId equal to 0 that is not a RASL, RADL or SLNR picture.

concatenation_flag indicates, when the current picture is not the first picture in the bitstream in decoding order, whether the nominal CPB removal time of the current picture is determined relative to the nominal CPB removal time of the preceding picture with a buffering period SEI message or relative to the nominal CPB removal time of the picture prevNonDiscardablePic.

au_cpb_removal_delay_delta_minus1 plus 1, when the current picture is not the first picture in the bitstream in decoding order, specifies a CPB removal delay increment value relative to the nominal CPB removal time of the picture prevNonDiscardablePic. This syntax element has a length in bits given by au_cpb_removal_delay_length_minus1 + 1.

When the current picture contains a buffering period SEI message and concatenation_flag is equal to 0 and the current picture is not the first picture in the bitstream in decoding order, it is a requirement of bitstream conformance that the following constraint applies:

– If the picture prevNonDiscardablePic is not associated with a buffering period SEI message, the au_cpb_removal_delay_minus1 of the current picture shall be equal to the au_cpb_removal_delay_minus1 of prevNonDiscardablePic plus au_cpb_removal_delay_delta_minus1 + 1.
– Otherwise, au_cpb_removal_delay_minus1 shall be equal to au_cpb_removal_delay_delta_minus1.

NOTE 3 – When the current picture contains a buffering period SEI message and concatenation_flag is equal to 1, the au_cpb_removal_delay_minus1 for the current picture is not used. The above-specified constraint can, under some circumstances, make it possible to splice bitstreams (that use suitably-designed referencing structures) by simply changing the value of concatenation_flag from 0 to 1 in the buffering period SEI message for an IRAP picture at the splicing point. When

concatenation_flag is equal to 0, the above-specified constraint enables the decoder to check whether the constraint is satisfied as a way to detect the loss of the picture prevNonDiscardablePic.

nal_initial_cpb_removal_delay[i] and **nal_initial_alt_cpb_removal_delay**[i] specify the default and the alternative initial CPB removal delays, respectively, for the i-th CPB when the NAL HRD parameters are in use. The syntax elements have a length in bits given by initial_cpb_removal_delay_length_minus1 + 1, and are in units of a 90 kHz clock. The values of the syntax elements shall not be equal to 0 and shall be less than or equal to 90 000 * (CpbSize[i] ÷ BitRate[i]), the time-equivalent of the CPB size in 90 kHz clock units.

nal_initial_cpb_removal_offset[i] and **nal_initial_alt_cpb_removal_offset**[i] specify the default and the alternative initial CPB removal offsets, respectively, for the i-th CPB when the NAL HRD parameters are in use. The syntax elements have a length in bits given by initial_cpb_removal_delay_length_minus1 + 1 and are in units of a 90 kHz clock.

Over the entire CVS, the sum of nal_initial_cpb_removal_delay[i] and nal_initial_cpb_removal_offset[i] shall be constant for each value of i, and the sum of nal_initial_alt_cpb_removal_delay[i] and nal_initial_alt_cpb_removal_offset[i] shall be constant for each value of i.

vcl_initial_cpb_removal_delay[i] and **vcl_initial_alt_cpb_removal_delay**[i] specify the default and the alternative initial CPB removal delays, respectively, for the i-th CPB when the VCL HRD parameters are in use. The syntax elements have a length in bits given by initial_cpb_removal_delay_length_minus1 + 1, and are in units of a 90 kHz clock. The values of the syntax elements shall not be equal to 0 and shall be less than or equal to 90 000 * (CpbSize[i] ÷ BitRate[i]), the time-equivalent of the CPB size in 90 kHz clock units.

vcl_initial_cpb_removal_offset[i] and **vcl_initial_alt_cpb_removal_offset**[i] specify the default and the alternative initial CPB removal offsets, respectively, for the i-th CPB when the VCL HRD parameters are in use. The syntax elements have a length in bits given by initial_cpb_removal_delay_length_minus1 + 1 and are in units of a 90 kHz clock.

Over the entire CVS, the sum of vcl_initial_cpb_removal_delay[i] and vcl_initial_cpb_removal_offset[i] shall be constant for each value of i, and the sum of vcl_initial_alt_cpb_removal_delay[i] and vcl_initial_alt_cpb_removal_offset[i] shall be constant for each value of i.

NOTE 4 – Encoders are recommended not to include irap_cpb_params_present_flag equal to 1 in buffering period SEI messages associated with a CRA or BLA picture for which at least one of its associated RASL pictures follows one or more of its associated RADL pictures in decoding order.

use_alt_cpb_params_flag may be used to derive the value of UseAltCpbParamsFlag. When irap_cpb_params_present_flag is equal to 0, use_alt_cpb_params_flag shall not be equal to 1. When use_alt_cpb_params_flag is not present, it is inferred to be equal to 0.

NOTE 5 – The syntax element use_alt_cpb_params_flag may be present in the payload extension of the buffering period SEI message. Decoders conforming to profiles specified in Annex A may ignore this syntax element.

It is a requirement of bitstream conformance that when use_alt_cpb_params_flag is present in the buffering perid SEI message, the return value of the more_data_in_payload( ) function in the sei_payload( ) syntax structure containing the buffering period SEI message shall be equal to 1.

D.3.3 Picture timing SEI message semantics

The picture timing SEI message provides CPB removal delay and DPB output delay information for the access unit associated with the SEI message.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, the following applies for the picture timing SEI message syntax and semantics:

– The syntax elements sub_pic_hrd_params_present_flag, sub_pic_cpb_params_in_pic_timing_sei_flag, au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1, dpb_output_delay_du_length_minus1, du_cpb_removal_delay_increment_length_minus1 and the variable CpbDpbDelaysPresentFlag are found in or derived from syntax elements found in the hrd_parameters( ) syntax structure that is applicable to at least one of the operation points to which the picture timing SEI message applies.

– The bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with any of the operation points to which the picture timing SEI message applies.

NOTE 1 – The syntax of the picture timing SEI message is dependent on the content of the hrd_parameters( ) syntax structures applicable to the operation points to which the picture timing SEI message applies. These hrd_parameters( ) syntax structures are in either or both of the VPS and the SPS that are active for the coded picture associated with the picture timing SEI message. When the picture timing SEI message is associated with an IRAP access unit with NoRaslOutputFlag equal to 1, unless it is preceded by an active parameter sets SEI message within the same access unit, the activation of the VPS and the SPS (and, for IRAP pictures with NoRaslOutputFlag equal to 1 that are not the first picture in the bitstream in decoding order, the determination that the coded picture is an IRAP access unit with NoRaslOutputFlag equal to 1) does not occur

until the decoding of the first coded slice segment NAL unit of the coded picture. Since the coded slice segment NAL unit of the coded picture follows the picture timing SEI message in NAL unit order, there may be cases in which it is necessary for a decoder to store the RBSP containing the picture timing SEI message until determining the active VPS and the active SPS for the coded picture, and then perform the parsing of the picture timing SEI message.

The presence of picture timing SEI messages for an operation point including the base layer is specified as follows:

– If frame_field_info_present_flag is equal to 1 or CpbDpbDelaysPresentFlag is equal to 1, a picture timing SEI message applicable to the operation point shall be associated with every access unit in the CVS.

– Otherwise, in the CVS there shall be no access unit that is associated with a picture timing SEI message applicable to the operation point.

When the picture timing SEI message is not scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, the semantics of pic_struct, source_scan_type and duplicate_flag apply to the picture with nuh_layer_id equal to 0 and are specified in the following paragraphs.

NOTE 2 – When the picture timing SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id greater than 0 or is contained in a bitstream partition nesting SEI message specified in Annex F, the semantics of pic_struct, source_scan_type and duplicate_flag are specified in Annex F. The frame-field information SEI message specified in Annex F can be used to indicate pic_struct, source_scan_type and duplicate_flag for non-base layers.

pic_struct indicates whether a picture should be displayed as a frame or as one or more fields and, for the display of frames when fixed_pic_rate_within_cvs_flag is equal to 1, may indicate a frame doubling or tripling repetition period for displays that use a fixed frame refresh interval equal to DpbOutputElementalInterval[n] as given by Equation E-76. The interpretation of pic_struct is specified in Table D.2. Values of pic_struct that are not listed in Table D.2 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore reserved values of pic_struct.

When present, it is a requirement of bitstream conformance that the value of pic_struct shall be constrained such that exactly one of the following conditions is true:

– The value of pic_struct is equal to 0, 7 or 8 for all pictures in the CVS.

– The value of pic_struct is equal to 1, 2, 9, 10, 11 or 12 for all pictures in the CVS.

– The value of pic_struct is equal to 3, 4, 5 or 6 for all pictures in the CVS.

When fixed_pic_rate_within_cvs_flag is equal to 1, frame doubling is indicated by pic_struct equal to 7, which indicates that the frame should be displayed two times consecutively on displays with a frame refresh interval equal to DpbOutputElementalInterval[n] as given by Equation E-76, and frame tripling is indicated by pic_struct equal to 8, which indicates that the frame should be displayed three times consecutively on displays with a frame refresh interval equal to DpbOutputElementalInterval[n] as given by Equation E-76.

NOTE 3 – Frame doubling can be used to facilitate the display, for example, of 25 Hz progressive-scan video on a 50 Hz progressive-scan display or 30 Hz progressive-scan video on a 60 Hz progressive-scan display. Using frame doubling and frame tripling in alternating combination on every other frame can be used to facilitate the display of 24 Hz progressive-scan video on a 60 Hz progressive-scan display.

The nominal vertical and horizontal sampling locations of samples in top and bottom fields for 4:2:0, 4:2:2 and 4:4:4 chroma formats are shown in Figure D.1, Figure D.2 and Figure D.3, respectively.

Association indicators for fields (pic_struct equal to 9 through 12) provide hints to associate fields of complementary parity together as frames. The parity of a field can be top or bottom, and the parity of two fields is considered complementary when the parity of one field is top and the parity of the other field is bottom.

When frame_field_info_present_flag is equal to 1, it is a requirement of bitstream conformance that the constraints specified in the third column of Table D.2 shall apply.

NOTE 4 – When frame_field_info_present_flag is equal to 0, then in many cases default values may be inferred or indicated by other means. In the absence of other indications of the intended display type of a picture, the decoder should infer the value of pic_struct as equal to 0 when frame_field_info_present_flag is equal to 0.

source_scan_type equal to 1 indicates that the source scan type of the associated picture should be interpreted as progressive. source_scan_type equal to 0 indicates that the source scan type of the associated picture should be interpreted as interlaced. source_scan_type equal to 2 indicates that the source scan type of the associated picture is unknown or unspecified. source_scan_type equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall interpret the value 3 for source_scan_type as equivalent to the value 2.

The following applies to the semantics of source_scan_type:

– If general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 1, the value of source_scan_type shall be equal to 0 when present, and should be inferred to be equal to 0 when not present.

– Otherwise, if general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 0, the value of source_scan_type shall be equal to 1 when present and should be inferred to be equal to 1 when not present.

– Otherwise, when general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 0, the value of source_scan_type shall be equal to 2 when present and should be inferred to be equal to 2 when not present.

duplicate_flag equal to 1 indicates that the current picture is indicated to be a duplicate of a previous picture in output order. duplicate_flag equal to 0 indicates that the current picture is not indicated to be a duplicate of a previous picture in output order.

NOTE 5 – The duplicate_flag should be used to mark coded pictures known to have originated from a repetition process such as 3:2 pull-down or other such duplication and picture rate interpolation methods. This flag would commonly be used when a video feed is encoded as a field sequence in a "transport pass-through" fashion, with known duplicate pictures tagged by setting duplicate_flag equal to 1.

NOTE 6 – When field_seq_flag is equal to 1 and duplicate_flag is equal to 1, this should be interpreted as an indication that the access unit contains a duplicated field of the previous field in output order with the same parity as the current field unless a pairing is otherwise indicated by the use of a pic_struct value in the range of 9 to 12, inclusive.

Table D.2 – Interpretation of pic_struct

Value	Indicated display of picture	Restrictions
0	(progressive) Frame	field_seq_flag shall be equal to 0
1	Top field	field_seq_flag shall be equal to 1
2	Bottom field	field_seq_flag shall be equal to 1
3	Top field, bottom field, in that order	field_seq_flag shall be equal to 0
4	Bottom field, top field, in that order	field_seq_flag shall be equal to 0
5	Top field, bottom field, top field repeated, in that order	field_seq_flag shall be equal to 0
6	Bottom field, top field, bottom field repeated, in that order	field_seq_flag shall be equal to 0
7	Frame doubling	field_seq_flag shall be equal to 0 fixed_pic_rate_within_cvs_flag shall be equal to 1
8	Frame tripling	field_seq_flag shall be equal to 0 fixed_pic_rate_within_cvs_flag shall be equal to 1
9	Top field paired with previous bottom field in output order	field_seq_flag shall be equal to 1
10	Bottom field paired with previous top field in output order	field_seq_flag shall be equal to 1
11	Top field paired with next bottom field in output order	field_seq_flag shall be equal to 1
12	Bottom field paired with next top field in output order	field_seq_flag shall be equal to 1

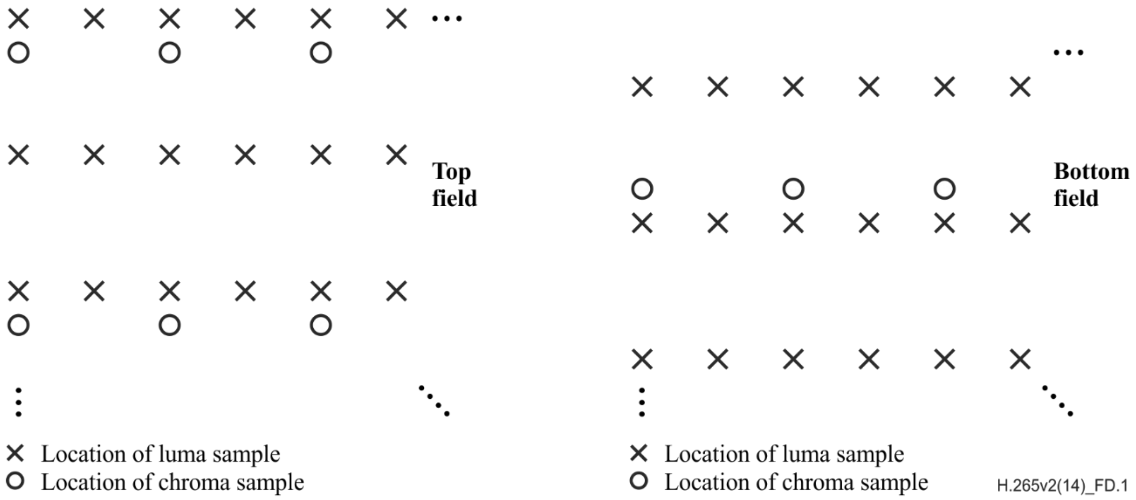


Figure D.1 – Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields

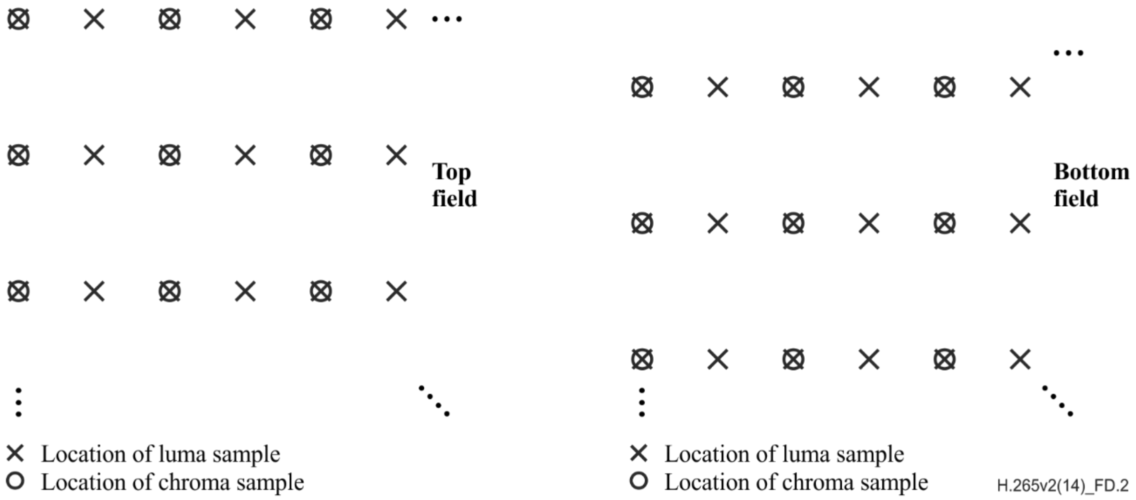


Figure D.2 – Nominal vertical and horizontal sampling locations of 4:2:2 samples in top and bottom fields

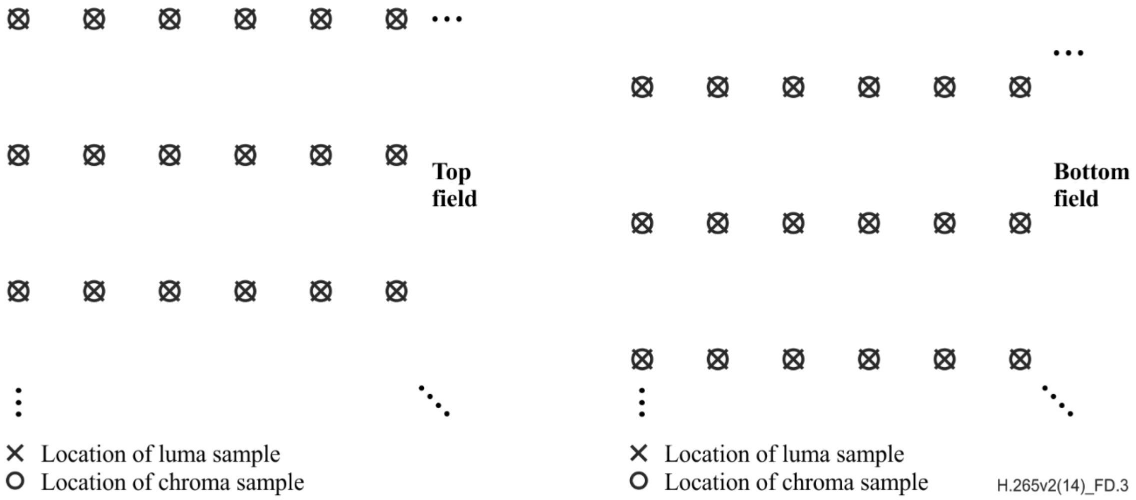


Figure D.3 – Nominal vertical and horizontal sampling locations of 4:4:4 samples in top and bottom fields

au_cpb_removal_delay_minus1 plus 1 is used to calculate the number of clock ticks between the nominal CPB removal times of the access unit associated with the picture timing SEI message and the preceding access unit in decoding order

that contained a buffering period SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS. The syntax element is a fixed length code whose length in bits is given by au_cpb_removal_delay_length_minus1 + 1.

NOTE 7 – The value of au_cpb_removal_delay_length_minus1 that determines the length (in bits) of the syntax element au_cpb_removal_delay_minus1 is the value of au_cpb_removal_delay_length_minus1 coded in the VPS or the SPS that is active for the coded picture associated with the picture timing SEI message, although the preceding access unit containing a buffering period SEI message may be an access unit of a different CVS.

The variable BpResetFlag of the current picture is derived as follows:

– If the current picture is associated with a buffering period SEI message that is applicable to at least one of the operation points to which the picture timing SEI message applies, BpResetFlag is set equal to 1.

– Otherwise, BpResetFlag is set equal to 0.

The variables AuCpbRemovalDelayMsb and AuCpbRemovalDelayVal of the current picture are derived as follows:

– If the current access unit is the access unit that initializes the HRD, AuCpbRemovalDelayMsb and AuCpbRemovalDelayVal are both set equal to 0.

– Otherwise, let the picture prevNonDiscardablePic be the previous picture in decoding order that has TemporalId equal to 0 that is not a RASL, RADL or SLNR picture, let prevAuCpbRemovalDelayMinus1, prevAuCpbRemovalDelayMsb and prevBpResetFlag be set equal to the values of au_cpb_removal_delay_minus1, AuCpbRemovalDelayMsb and BpResetFlag, respectively, for the picture prevNonDiscardablePic, and the following applies:

  – AuCpbRemovalDelayMsb is derived as follows:

```
if( prevBpResetFlag )
   AuCpbRemovalDelayMsb = 0
else if( au_cpb_removal_delay_minus1  <=  prevAuCpbRemovalDelayMinus1 )
   AuCpbRemovalDelayMsb = prevAuCpbRemovalDelayMsb + 2^(au_cpb_removal_delay_length_minus1 + 1)
   (D-1)
else
   AuCpbRemovalDelayMsb = prevAuCpbRemovalDelayMsb
```

  – AuCpbRemovalDelayVal is derived as follows:

$$\text{AuCpbRemovalDelayVal} = \text{AuCpbRemovalDelayMsb} + \text{au\_cpb\_removal\_delay\_minus1} + 1 \quad \text{(D-2)}$$

The value of AuCpbRemovalDelayVal shall be in the range of 1 to 2^{32}, inclusive.

pic_dpb_output_delay is used to compute the DPB output time of the picture when SubPicHrdFlag is equal to 0. It specifies how many clock ticks to wait after removal of the last decoding unit in an access unit from the CPB before the decoded picture is output from the DPB.

NOTE 8 – A picture is not removed from the DPB at its output time when it is still marked as "used for short-term reference" or "used for long-term reference".

The length of the syntax element pic_dpb_output_delay is given in bits by dpb_output_delay_length_minus1 + 1. When sps_max_dec_pic_buffering_minus1[minTid] is equal to 0, where minTid is the minimum of the OpTid values of all operation points the picture timing SEI message applies to, pic_dpb_output_delay shall be equal to 0.

The output time derived from the pic_dpb_output_delay of any picture that is output from an output timing conforming decoder shall precede the output time derived from the pic_dpb_output_delay of all pictures in any subsequent CVS in decoding order.

The picture output order established by the values of this syntax element shall be the same order as established by the values of PicOrderCntVal.

For pictures that are not output by the "bumping" process because they precede, in decoding order, an IRAP picture with NoRaslOutputFlag equal to 1 that has no_output_of_prior_pics_flag equal to 1 or inferred to be equal to 1, the output times derived from pic_dpb_output_delay shall be increasing with increasing value of PicOrderCntVal relative to all pictures within the same CVS.

pic_dpb_output_du_delay is used to compute the DPB output time of the picture when SubPicHrdFlag is equal to 1. It specifies how many sub clock ticks to wait after removal of the last decoding unit in an access unit from the CPB before the decoded picture is output from the DPB.

The length of the syntax element pic_dpb_output_du_delay is given in bits by dpb_output_delay_du_length_minus1 + 1.

The output time derived from the pic_dpb_output_du_delay of any picture that is output from an output timing conforming decoder shall precede the output time derived from the pic_dpb_output_du_delay of all pictures in any subsequent CVS in decoding order.

The picture output order established by the values of this syntax element shall be the same order as established by the values of PicOrderCntVal.

For pictures that are not output by the "bumping" process because they precede, in decoding order, an IRAP picture with NoRaslOutputFlag equal to 1 that has no_output_of_prior_pics_flag equal to 1 or inferred to be equal to 1, the output times derived from pic_dpb_output_du_delay shall be increasing with increasing value of PicOrderCntVal relative to all pictures within the same CVS.

For any two pictures in the CVS, the difference between the output times of the two pictures when SubPicHrdFlag is equal to 1 shall be identical to the same difference when SubPicHrdFlag is equal to 0.

num_decoding_units_minus1 plus 1 specifies the number of decoding units in the access unit the picture timing SEI message is associated with. The value of num_decoding_units_minus1 shall be in the range of 0 to PicSizeInCtbsY − 1, inclusive.

du_common_cpb_removal_delay_flag equal to 1 specifies that the syntax element du_common_cpb_removal_delay_increment_minus1 is present. du_common_cpb_removal_delay_flag equal to 0 specifies that the syntax element du_common_cpb_removal_delay_increment_minus1 is not present.

du_common_cpb_removal_delay_increment_minus1 plus 1 specifies the duration, in units of clock sub-ticks (see clause E.3.2), between the nominal CPB removal times of any two consecutive decoding units in decoding order in the access unit associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of decoding unit data into the CPB for the HSS, as specified in Annex C or clause F.13. The syntax element is a fixed length code whose length in bits is given by du_cpb_removal_delay_increment_length_minus1 + 1.

num_nalus_in_du_minus1[i] plus 1 specifies the number of NAL units in the i-th decoding unit of the access unit the picture timing SEI message is associated with. The value of num_nalus_in_du_minus1[i] shall be in the range of 0 to PicSizeInCtbsY − 1, inclusive.

The first decoding unit of the access unit consists of the first num_nalus_in_du_minus1[0] + 1 consecutive NAL units in decoding order in the access unit. The i-th (with i greater than 0) decoding unit of the access unit consists of the num_nalus_in_du_minus1[i] + 1 consecutive NAL units immediately following the last NAL unit in the previous decoding unit of the access unit, in decoding order. There shall be at least one VCL NAL unit in each decoding unit. All non-VCL NAL units associated with a VCL NAL unit shall be included in the same decoding unit as the VCL NAL unit.

du_cpb_removal_delay_increment_minus1[i] plus 1 specifies the duration, in units of clock sub-ticks, between the nominal CPB removal times of the (i + 1)-th decoding unit and the i-th decoding unit, in decoding order, in the access unit associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of decoding unit data into the CPB for the HSS, as specified in Annex C or clause F.13. The syntax element is a fixed length code whose length in bits is given by du_cpb_removal_delay_increment_length_minus1 + 1.

D.3.4 Pan-scan rectangle SEI message semantics

The pan-scan rectangle SEI message syntax elements specify the coordinates of one or more rectangles relative to the conformance cropping window specified by the active SPS. Each coordinate is specified in units of one-sixteenth luma sample spacing relative to the luma sampling grid.

pan_scan_rect_id contains an identifying number that may be used to identify the purpose of the one or more pan-scan rectangles (for example, to identify the one or more rectangles as the area to be shown on a particular display device or as the area that contains a particular actor in the scene). The value of pan_scan_rect_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of pan_scan_rect_id from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of pan_scan_rect_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of pan_scan_rect_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

pan_scan_rect_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous pan-scan rectangle SEI message in output order that applies to the current layer. pan_scan_rect_cancel_flag equal to 0 indicates that pan-scan rectangle information follows.

pan_scan_cnt_minus1 specifies the number of pan-scan rectangles that are specified by the SEI message. pan_scan_cnt_minus1 shall be in the range of 0 to 2, inclusive.

pan_scan_cnt_minus1 equal to 0 indicates that a single pan-scan rectangle is specified that applies to the decoded pictures that are within the persistence scope of the current SEI message. When field_seq_flag is equal to 1, pan_scan_cnt_minus1 shall be equal to 0.

pan_scan_cnt_minus1 equal to 1 indicates that two pan-scan rectangles are specified that apply to the decoded pictures that are within the persistence scope of the current SEI message and that are associated with picture timing SEI messages having pic_struct equal to 3 or 4. The first rectangle applies to the first field of a frame in output order and the second rectangle applies to the second field of a frame in output order, where the output order between two fields in one frame is as shown in Table D.2 for pic_struct equal to 3 or 4.

pan_scan_cnt_minus1 equal to 2 indicates that three pan-scan rectangles are specified that apply to the decoded pictures that are within the persistence scope of the current SEI message and that are associated with picture timing SEI messages having pic_struct equal to 5 or 6. The first rectangle applies to the first field of the frame in output order, the second rectangle applies to the second field of the frame in output order, and the third rectangle applies to a repetition of the first field as a third field in output order, where the output order of fields in one frame is as shown in Table D.2 for pic_struct equal to 5 or 6.

pan_scan_rect_left_offset[i], **pan_scan_rect_right_offset**[i], **pan_scan_rect_top_offset**[i] and **pan_scan_rect_bottom_offset**[i], specify, as signed integer quantities in units of one-sixteenth sample spacing relative to the luma sampling grid, the location of the i-th pan-scan rectangle. The values of each of these four syntax elements shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive.

The pan-scan rectangle is specified, in units of one-sixteenth sample spacing relative to a luma sampling grid, as the region with horizontal coordinates from 16 * SubWidthC * conf_win_left_offset + pan_scan_rect_left_offset[i] to 16 * (CtbSizeY * PicWidthInCtbsY − SubWidthC * conf_win_right_offset) + pan_scan_rect_right_offset[i] − 1 and with vertical coordinates from 16 *SubHeightC * conf_win_top_offset + pan_scan_rect_top_offset[i] to 16 * (CtbSizeY * PicHeightInCtbsY − SubHeightC * conf_win_bottom_offset) + pan_scan_rect_bottom_offset[i] − 1, inclusive. The value of 16 * SubWidthC * conf_win_left_offset + pan_scan_rect_left_offset[i] shall be less than or equal to 16 * (CtbSizeY * PicWidthInCtbsY − SubWidthC * conf_win_right_offset) + pan_scan_rect_right_offset[i] − 1 and the value of 16 * SubHeightC * conf_win_top_offset + pan_scan_rect_top_offset[i] shall be less than or equal to 16 * (CtbSizeY * PicHeightInCtbsY − SubHeightC * conf_win_bottom_offset) + pan_scan_rect_bottom_offset[i] − 1.

When the pan-scan rectangular area includes samples outside of the conformance cropping window, the region outside of the conformance cropping window may be filled with synthesized content (such as black video content or neutral grey video content) for display.

pan_scan_rect_persistence_flag specifies the persistence of the pan-scan rectangle SEI message for the current layer.

pan_scan_rect_persistence_flag equal to 0 specifies that the pan-scan rectangle information applies to the current decoded picture only.

Let picA be the current picture. pan_scan_rect_persistence_flag equal to 1 specifies that the pan-scan rectangle information persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.
– The bitstream ends.
– A picture picB in the current layer in an access unit containing a pan-scan rectangle SEI message with the same value of pan_scan_rect_id and applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

D.3.5 Filler payload SEI message semantics

This SEI message contains a series of payloadSize bytes of value 0xFF, which can be discarded.

ff_byte shall be a byte having the value 0xFF.

D.3.6 User data registered by Recommendation ITU-T T.35 SEI message semantics

This SEI message contains user data registered as specified in Recommendation ITU-T T.35, the contents of which are not specified in this Specification.

itu_t_t35_country_code shall be a byte having a value specified as a country code by Annex A of Recommendation ITU-T T.35.

itu_t_t35_country_code_extension_byte shall be a byte having a value specified as a country code by Annex B of Recommendation ITU-T T.35.

itu_t_t35_payload_byte shall be a byte containing data registered as specified in Recommendation ITU-T T.35.

The ITU-T T.35 terminal provider code and terminal provider oriented code shall be contained in the first one or more bytes of the itu_t_t35_payload_byte, in the format specified by the Administration that issued the terminal provider code. Any remaining itu_t_t35_payload_byte data shall be data having syntax and semantics as specified by the entity identified by the ITU-T T.35 country code and terminal provider code.

D.3.7 User data unregistered SEI message semantics

This SEI message contains unregistered user data identified by a universal unique identifier (UUID), the contents of which are not specified in this Specification.

uuid_iso_iec_11578 shall have a value specified as a UUID according to the procedures of Annex A of ISO/IEC 11578:1996.

user_data_payload_byte shall be a byte containing data having syntax and semantics as specified by the UUID generator.

D.3.8 Recovery point SEI message semantics

The recovery point SEI message assists a decoder in determining when the decoding process will produce acceptable pictures for display after the decoder initiates random access or after the encoder indicates a broken link in the CVS. When the decoding process is started with the access unit in decoding order associated with the recovery point SEI message, all decoded pictures at or subsequent to the recovery point in output order specified in this SEI message are indicated to be correct or approximately correct in content. Decoded pictures produced by random access at or before the picture associated with the recovery point SEI message need not be correct in content until the indicated recovery point, and the operation of the decoding process starting at the picture associated with the recovery point SEI message may contain references to pictures unavailable in the decoded picture buffer.

In addition, by use of the broken_link_flag, the recovery point SEI message can indicate to the decoder the location of some pictures in the bitstream that can result in serious visual artefacts when displayed, even when the decoding process was begun at the location of a previous IRAP access unit in decoding order.

NOTE 1 – The broken_link_flag can be used by encoders to indicate the location of a point after which the decoding process for the decoding of some pictures may cause references to pictures that, though available for use in the decoding process, are not the pictures that were used for reference when the bitstream was originally encoded (e.g., due to a splicing operation performed during the generation of the bitstream).

When random access is performed to start decoding from the access unit associated with the recovery point SEI message, the decoder operates as if the associated picture was the first picture in the bitstream in decoding order, and the variables prevPicOrderCntLsb and prevPicOrderCntMsb used in derivation of PicOrderCntVal are both set equal to 0.

NOTE 2 – When HRD information is present in the bitstream, a buffering period SEI message should be associated with the access unit associated with the recovery point SEI message in order to establish initialization of the HRD buffer model after a random access.

Any SPS or PPS RBSP that is referred to by a picture associated with a recovery point SEI message or by any picture following such a picture in decoding order shall be available to the decoding process prior to its activation, regardless of whether or not the decoding process is started at the beginning of the bitstream or with the access unit, in decoding order, that is associated with the recovery point SEI message.

recovery_poc_cnt specifies the recovery point of decoded pictures in output order. If there is a picture picA that follows the current picture (i.e., the picture associated with the current SEI message) in decoding order in the CVS and that has PicOrderCntVal equal to the PicOrderCntVal of the current picture plus the value of recovery_poc_cnt, the picture picA is referred to as the recovery point picture. Otherwise, the first picture in output order that has PicOrderCntVal greater than the PicOrderCntVal of the current picture plus the value of recovery_poc_cnt is referred to as the recovery point picture. The recovery point picture shall not precede the current picture in decoding order. All decoded pictures in output order are indicated to be correct or approximately correct in content starting at the output order position of the recovery point picture. The value of recovery_poc_cnt shall be in the range of $-$MaxPicOrderCntLsb / 2 to MaxPicOrderCntLsb / 2 $-$ 1, inclusive.

exact_match_flag indicates whether decoded pictures at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit associated with the recovery point SEI message will be an exact match to the pictures that would be produced by starting the decoding process at the location of a previous IRAP access unit, if any, in the bitstream. The value 0 indicates that the match may not be exact and the value 1 indicates that the match will be exact. When exact_match_flag is equal to 1, it is a requirement of bitstream conformance that the decoded pictures at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit associated with the recovery point SEI message shall be an exact match to the pictures that would be produced by starting the decoding process at the location of a previous IRAP access unit, if any, in the bitstream.

NOTE 3 – When performing random access, decoders should infer all references to unavailable pictures as references to pictures containing only intra coding blocks and having sample values given by Y equal to $(1 \ll (\mathrm{BitDepth}_Y - 1))$, Cb and Cr both equal to $(1 \ll (\mathrm{BitDepth}_C - 1))$ (mid-level grey), regardless of the value of exact_match_flag.

When exact_match_flag is equal to 0, the quality of the approximation at the recovery point is chosen by the encoding process and is not specified in this Specification.

broken_link_flag indicates the presence or absence of a broken link in the NAL unit stream at the location of the recovery point SEI message and is assigned further semantics as follows:

– If broken_link_flag is equal to 1, pictures produced by starting the decoding process at the location of a previous IRAP access unit may contain undesirable visual artefacts to the extent that decoded pictures at and subsequent to the access unit associated with the recovery point SEI message in decoding order should not be displayed until the specified recovery point in output order.
– Otherwise (broken_link_flag is equal to 0), no indication is given regarding any potential presence of visual artefacts.

When the current picture is a BLA picture, the value of broken_link_flag shall be equal to 1.

Regardless of the value of the broken_link_flag, pictures subsequent to the specified recovery point in output order are specified to be correct or approximately correct in content.

D.3.9 Scene information SEI message semantics

A scene and a scene transition are herein defined as a set of consecutive pictures in output order.

NOTE 1 – Decoded pictures within one scene generally have similar content. The scene information SEI message is used to label pictures with scene identifiers and to indicate scene changes. The message specifies how the source pictures for the labelled pictures were created. The decoder may use the information to select an appropriate algorithm to conceal transmission errors. For example, a specific algorithm may be used to conceal transmission errors that occurred in pictures belonging to a gradual scene transition. Furthermore, the scene information SEI message may be used in a manner determined by the application, such as for indexing the scenes of a video sequence.

A scene information SEI message labels all pictures of the current layer, in decoding order, from the coded picture to which the SEI message is associated (inclusive) to the coded picture to which the next scene information SEI message applicable to the current layer (when present) in decoding order is associated (exclusive) or (otherwise) to the last picture in the CLVS (inclusive). These pictures are herein referred to as the target pictures.

NOTE 2 – The semantics of the scene information SEI message apply layer-wise. However, the scene information SEI message may be contained within a scalable nesting SEI message, which may help in reducing the number of scene information SEI messages, as scene changes and transitions apply across layers.

scene_info_present_flag equal to 0 indicates that the scene or scene transition to which the target pictures belong is unspecified. scene_info_present_flag equal to 1 indicates that the target pictures belong to the same scene or scene transition.

prev_scene_id_valid_flag equal to 0 specifies that the scene_id value of the picture preceding the first picture of the target pictures in output order is considered unspecified in the semantics of the syntax elements of this SEI message. prev_scene_id_valid_flag equal to 1 specifies that the scene_id value of the picture preceding the first picture of the target pictures in output order is specified by the previous scene information SEI message in decoding order. When the previous scene information SEI message applicable to the current layer is within the same CLVS as the current scene information SEI message, prev_scene_id_valid_flag shall be equal to 1.

NOTE 3 – When a current scene information SEI message is associated with the first picture, in decoding order, of a CLVS, prev_scene_id_valid_flag equal to 1 indicates that the scene_id values of the current scene information SEI message and the previous scene information SEI message applicable to the current layer in decoding order can be used to conclude whether their target pictures belong to the same scene or to different scenes.

NOTE 4 – When CVS B is concatenated to CVS A and CVS A represents a different scene than the scene CVS B represents, it should be noticed that the scene_id value specified for the last picture with a particular nuh_layer_id value of CVS A affects the semantics of the scene information SEI message associated with that particular nuh_layer_id value and the first picture, in decoding order, of CVS B, when the SEI message is present. Hence, as part of such a concatenation operation, the value of prev_scene_id_valid_flag should be set equal to 0 in the scene information SEI message associated with the first picture, in decoding order, of CVS B, when the SEI message is present.

scene_id identifies the scene to which the target pictures belong. When the value of scene_transition_type of the target pictures is less than 4, and the previous picture in output order is marked with a value of scene_transition_type less than 4, and the value of scene_id is the same as the value of scene_id of the previous picture in output order, this indicates that the source scene for the target pictures and the source scene for the previous picture (in output order) are considered by the encoder to have been the same scene. When the value of scene_transition_type of the target pictures is greater than 3, and the previous picture in output order is marked with a value of scene_transition_type less than 4, and the value of scene_id is the same as the value of scene_id of the previous picture in output order, this indicates that one of the source scenes for the target pictures and the source scene for the previous picture (in output order) are considered by the encoder to have been the same scene. When the value of scene_id is not equal to the value of scene_id of the previous picture in output order, this indicates that the target pictures and the previous picture (in output order) are considered by the encoder to have been from different source scenes.

The value of scene_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of scene_id in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of scene_id in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of scene_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 5 – When the first picture picA, in decoding order, of the CLVS vidSeqA represents a different scene than the last picture, in output order, of the previous CLVS of the same layer and a scene information SEI message is associated with PicA, the scene_id value of that scene information SEI message should have a random value within the value ranges constrained above. Subsequent scene_id and second_scene_id values may be selected for example by incrementing the initial randomly selected scene_id value. Consequently, when concatenating vidSeqA to a CLVS vidSeqB of the same layer, accidental use of the same scene_id values in vidSeqA and vidSeqB is unlikely.

scene_transition_type specifies in which type of a scene transition (if any) the target pictures are involved. The valid values of scene_transition_type are specified in Table D.3.

Table D.3 – scene_transition_type values

Value	Description
0	No transition
1	Fade to black
2	Fade from black
3	Unspecified transition from or to constant colour
4	Dissolve
5	Wipe
6	Unspecified mixture of two scenes

When scene_transition_type is greater than 3, the target pictures include contents both from the scene labelled by its scene_id and the next scene, in output order, which is labelled by second_scene_id (see below). The term "the current scene" is used to indicate the scene labelled by scene_id. The term "the next scene" is used to indicate the scene labelled by second_scene_id. It is not required for any following picture, in output order, to be labelled with scene_id equal to second_scene_id of the current SEI message.

Scene transition types are specified as follows:

– "No transition" specifies that the target pictures are not involved in a gradual scene transition.

  NOTE 6 – When two consecutive pictures in output order have scene_transition_type equal to 0 and different values of scene_id, a scene cut occurred between the two pictures.

– "Fade to black" indicates that the target pictures are part of a sequence of pictures, in output order, involved in a fade to black scene transition, i.e., the luma samples of the scene gradually approach zero and the chroma samples of the scene gradually approach 128.

  NOTE 7 – When two pictures are labelled to belong to the same scene transition and their scene_transition_type is "Fade to black", the later one, in output order, is darker than the previous one.

– "Fade from black" indicates that the target pictures are part of a sequence of pictures, in output order, involved in a fade from black scene transition, i.e., the luma samples of the scene gradually diverge from zero and the chroma samples of the scene may gradually diverge from 128.

  NOTE 8 – When two pictures are labelled to belong to the same scene transition and their scene_transition_type is "Fade from black", the later one in output order is lighter than the previous one.

– "Dissolve" indicates that the sample values of each target picture (before encoding) were generated by calculating a sum of co-located weighted sample values of a picture from the current scene and a picture from the next scene. The weight of the current scene gradually decreases from full level to zero level, whereas the weight of the next scene gradually increases from zero level to full level. When two pictures are labelled to belong to the same scene transition and their scene_transition_type is "Dissolve", the weight of the current scene for the later one, in output order, is less than the weight of the current scene for the previous one, and the weight of the next scene for the later one, in output order, is greater than the weight of the next scene for the previous one.

– "Wipe" indicates that some of the sample values of each target picture (before encoding) were generated by copying co-located sample values of a picture in the current scene and the remaining sample values of each target picture

(before encoding) were generated by copying co-located sample values of a picture in the next scene. When two pictures are labelled to belong to the same scene transition and their scene_transition_type is "Wipe", the number of samples copied from the next scene to the later picture in output order is greater than the number of samples copied from the next scene to the previous picture.

second_scene_id identifies the next scene in the gradual scene transition in which the target pictures are involved. The value of second_scene_id shall not be equal to the value of scene_id. The value of second_scene_id shall not be equal to the value of scene_id in the previous picture in output order. When the next picture in output order is marked with a value of scene_transition_type less than 4, and the value of second_scene_id is the same as the value of scene_id of the next picture in output order, this indicates that the encoder considers one of the source scenes for the target pictures and the source scene for the next picture (in output order) to have been the same scene. When the value of second_scene_id is not equal to the value of scene_id or second_scene_id (when present) of the next picture in output order, this indicates that the encoder considers the target pictures and the next picture (in output order) to have been from different source scenes.

When the value of scene_id of a picture is equal to the value of scene_id of the following picture in output order and the value of scene_transition_type in both of these pictures is less than 4, this indicates that the encoder considers the two pictures to have been from the same source scene. When the values of scene_id, scene_transition_type and second_scene_id (when present) of a picture are equal to the values of scene_id, scene_transition_type and second_scene_id (respectively) of the following picture in output order and the value of scene_transition_type is greater than 0, this indicates that the encoder considers the two pictures to have been from the same source gradual scene transition.

The value of second_scene_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of second_scene_id in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of second_scene_id in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of second_scene_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

D.3.10 Picture snapshot SEI message semantics

The picture snapshot SEI message indicates that the current picture is labelled for use as determined by the application as a still-image snapshot of the video content.

snapshot_id specifies a snapshot identification number. snapshot_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of snapshot_id in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of snapshot_id in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of snapshot_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

D.3.11 Progressive refinement segment start SEI message semantics

The progressive refinement segment start SEI message specifies the beginning of a set of consecutive coded pictures in the current layer in decoding order that consists of the current picture and a sequence of one or more subsequent pictures in the current layer that refine the quality of the current picture, rather than a representation of a continually moving scene.

Let picA be the current picture. The tagged set of consecutive coded pictures refinementPicSet in the current layer consists of, in decoding order, the next picture in the current layer after the current picture in decoding order, when present, followed by zero or more pictures in the current layer, including all subsequent pictures in the current layer up to but not including any subsequent picture picB in the current layer for which one of the following conditions is true:

– The picture picB starts a new CLVS of the current layer.

– The value of pic_order_cnt_delta is greater than 0 and the PicOrderCntVal of the picture picB, i.e., PicOrderCnt(picB) is greater than PicOrderCnt(picA) + pic_order_cnt_delta, where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

– The picture picB is associated with a progressive refinement segment end SEI message that has the same progressive_refinement_id as the one in this SEI message and also applies to the current layer is decoded.

The decoding order of pictures within refinementPicSet should be the same as their output order.

progressive_refinement_id specifies an identification number for the progressive refinement operation. progressive_refinement_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of progressive_refinement_id in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of progressive_refinement_id in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of progressive_refinement_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

pic_order_cnt_delta specifies the last picture in refinementPicSet in decoding order as follows:

- If pic_order_cnt_delta is equal to 0, the last picture in refinementPicSet in decoding order is the following picture:
  - If the CLVS contains one or more pictures in the current layer that follow the current picture in decoding order and are associated with a progressive refinement segment end SEI message that has the same progressive_refinement_id and also applies to the current layer, the last picture in refinementPicSet in decoding order is the first of these pictures in decoding order.
  - Otherwise, the last picture in refinementPicSet in decoding order is the last picture in the current layer within the CLVS in decoding order.
- Otherwise, the last picture in refinementPicSet in decoding order is the following picture:
  - If the CLVS contains one or more pictures in the current layer that follow the current picture in decoding order, that are associated with a progressive refinement segment end SEI message with the same progressive_refinement_id and applicable to the current layer, and that precede any picture picC in the current layer in the CLVS for which PicOrderCnt(picC) is greater than PicOrderCnt(picA) + pic_order_cnt_delta, where PicOrderCnt(picC) and PicOrderCnt(picA) are the PicOrderCntVal of the picC and picA, respectively, immediately after the invocation of the decoding process for picture order count for picC, the last picture in refinementPicSet in decoding order is the first of these pictures in decoding order.
  - Otherwise, if the CLVS contains one or more pictures picD in the current layer that follow the current picture in decoding order for which PicOrderCnt(picD) is greater than PicOrderCnt(picA) + pic_order_cnt_delta, where PicOrderCnt(picD) and PicOrderCnt(picA) are the PicOrderCntVal values of picD and picA, respectively, immediately after the invocation of the decoding process for picture order count for picD, the last picture in refinementPicSet in decoding order is the last picture in the current layer that precedes the first of these pictures in decoding order.
  - Otherwise, the last picture in refinementPicSet in decoding order is the last picture in the current layer within the CLVS in decoding order.

The value of pic_order_cnt_delta shall be in the range of 0 to 256, inclusive.

D.3.12 Progressive refinement segment end SEI message semantics

The progressive refinement segment end SEI message specifies the end of a set of consecutive coded pictures that has been labelled by use of a progressive refinement segment start SEI message as an initial picture followed by a sequence of one or more pictures of the refinement of the quality of the initial picture and ending with the current picture.

progressive_refinement_id specifies an identification number for the progressive refinement operation. progressive_refinement_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

The progressive refinement segment end SEI message specifies the end of any progressive refinement segment previously started using a progressive refinement segment start SEI message with the same value of progressive_refinement_id.

Values of progressive_refinement_id in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of progressive_refinement_id in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of progressive_refinement_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

D.3.13 Film grain characteristics SEI message semantics

This SEI message provides the decoder with a parameterized model for film grain synthesis.

NOTE 1 – For example, an encoder could use the film grain characteristics SEI message to characterize film grain that was present in the original source video material and was removed by pre-processing filtering techniques. Synthesis of simulated film grain on the decoded images for the display process is optional and does not need to exactly follow the specified semantics of the film grain characteristics SEI message. When synthesis of simulated film grain on the decoded images for the display process is performed, there is no requirement that the method by which the synthesis is performed be the same as the parameterized model for the film grain as provided in the film grain characteristics SEI message.

NOTE 2 – The display process is not specified in this Specification.

NOTE 3 – SMPTE RDD 5 specifies a film grain simulator based on the information provided in the film grain characteristics SEI message.

The film grain models specified in the film grain characteristics SEI message are expressed for application to decoded pictures that have 4:4:4 colour format with luma and chroma bit depths corresponding to the luma and chroma bit depths of the film grain model and use the same colour representation domain as the identified film grain model. When the colour format of the decoded video is not 4:4:4 or the decoded video uses a different luma or chroma bit depth from that of the film grain model or uses a different colour representation domain from that of the identified film grain model, an unspecified conversion process is expected to be applied to convert the decoded pictures to the form that is expressed for application of the film grain model.

NOTE 4 – Because the use of a specific method is not required for performing the film grain generation function used by the display process, a decoder could, if desired, down-convert the model information for chroma in order to simulate film grain for other chroma formats (4:2:0 or 4:2:2) rather than up-converting the decoded video (using a method not specified in this Specification) before performing film grain generation.

film_grain_characteristics_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous film grain characteristics SEI message in output order that applies to the current layer. film_grain_characteristics_cancel_flag equal to 0 indicates that film grain modelling information follows.

film_grain_model_id identifies the film grain simulation model as specified in Table D.4. The value of film_grain_model_id shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for film_grain_model_id are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore film grain characteristic SEI messages with film_grain_model_id equal to 2 or 3.

Table D.4 – film_grain_model_id values

Value	Description
0	Frequency filtering
1	Auto-regression

separate_colour_description_present_flag equal to 1 indicates that a distinct combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message is present in the film grain characteristics SEI message syntax. separate_colour_description_present_flag equal to 0 indicates that the combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message are the same as indicated in the VUI parameters for the CVS.

NOTE 5 – When separate_colour_description_present_flag is equal to 1, any of the luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients specified for the film grain characteristics specified in the SEI message could differ from those for the pictures in the CVS.

When VUI parameters are not present for the CVS or the value of colour_description_present_flag is equal to 0, and equivalent information to that conveyed when colour_description_present_flag is equal to 1 is not conveyed by external means, separate_colour_description_present_flag shall be equal to 1.

The decoded image $I_{decoded}$ used in the equations in this clause is in the same colour representation domain as the simulated film grain signal. Therefore, when any of these parameters does differ from that for the pictures in the CVS, the decoded image $I_{decoded}$ used in the equations in this clause would be in a different colour representation domain than that for the pictures in the CVS. For example, when the value of film_grain_bit_depth_luma_minus8 + 8 is greater than the bit depth of the luma component of the pictures in the CVS, the bit depth of $I_{decoded}$ used in the equations in this clause is also greater than the bit depth of the luma component of the pictures in the CVS. In such a case, the decoded image $I_{decoded}$ corresponding to an actual decoded picture would be generated by converting the actual decoded picture to be in the same colour representation domain as the simulated film grain signal. The process for converting the actual decoded pictures to the 4:4:4 colour format with same colour representation domain as the simulated film grain signal is not specified in this Specification.

film_grain_bit_depth_luma_minus8 plus 8 specifies the bit depth used for the luma component of the film grain characteristics specified in the SEI message. When film_grain_bit_depth_luma_minus8 is not present in the film grain characteristics SEI message, the value of film_grain_bit_depth_luma_minus8 is inferred to be equal to bit_depth_luma_minus8.

The value of filmGrainBitDepth[0] is derived as follows:

$$\text{filmGrainBitDepth[0] = film\_grain\_bit\_depth\_luma\_minus8 + 8} \quad \text{(D-3)}$$

film_grain_bit_depth_chroma_minus8 plus 8 specifies the bit depth used for the Cb and Cr components of the film grain characteristics specified in the SEI message. When film_grain_bit_depth_chroma_minus8 is not present in the film grain characteristics SEI message, the value of film_grain_bit_depth_chroma_minus8 is inferred to be equal to bit_depth_chroma_minus8.

The value of filmGrainBitDepth[c] for c = 1 and 2 is derived as follows:

$$\text{filmGrainBitDepth[c] = film\_grain\_bit\_depth\_chroma\_minus8 + 8, with c = 1, 2} \quad \text{(D-4)}$$

film_grain_full_range_flag has the same semantics as specified in clause E.3.1 for the video_full_range_flag syntax element, except as follows:

– film_grain_full_range_flag specifies the video full range flag of the film grain characteristics specified in the SEI message, rather than the video full range flag used for the CVS.

– When film_grain_full_range_flag is not present in the film grain characteristics SEI message, the value of film_grain_full_range_flag is inferred to be equal to video_full_range_flag.

film_grain_colour_primaries has the same semantics as specified in clause E.3.1 for the colour_primaries syntax element, except as follows:

– film_grain_colour_primaries specifies the colour primaries of the film grain characteristics specified in the SEI message, rather than the colour primaries used for the CVS.

– When film_grain_colour_primaries is not present in the film grain characteristics SEI message, the value of film_grain_colour_primaries is inferred to be equal to colour_primaries.

film_grain_transfer_characteristics has the same semantics as specified in clause E.3.1 for the transfer_characteristics syntax element, except as follows:

– film_grain_transfer_characteristics specifies the transfer characteristics of the film grain characteristics specified in the SEI message, rather than the transfer characteristics used for the CVS.

– When film_grain_transfer_characteristics is not present in the film grain characteristics SEI message, the value of film_grain_transfer_characteristics is inferred to be equal to transfer_characteristics.

film_grain_matrix_coeffs has the same semantics as specified in clause E.3.1 for the matrix_coeffs syntax element, except as follows:

– film_grain_matrix_coeffs specifies the matrix coefficients of the film grain characteristics specified in the SEI message, rather than the matrix coefficients used for the CVS.

– When film_grain_matrix_coeffs is not present in the film grain characteristics SEI message, the value of film_grain_matrix_coeffs is inferred to be equal to matrix_coeffs.

– The values allowed for film_grain_matrix_coeffs are not constrained by the chroma format of the decoded pictures that is indicated by the value of chroma_format_idc for the semantics of the VUI parameters.

blending_mode_id identifies the blending mode used to blend the simulated film grain with the decoded images as specified in Table D.5. blending_mode_id shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for blending_mode_id are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore film grain characteristic SEI messages with blending_mode_id equal to 2 or 3.

Table D.5 – blending_mode_id values

Value	Description
0	Additive
1	Multiplicative

Depending on the value of blending_mode_id, the blending mode is specified as follows:

– If blending_mode_id is equal to 0, the blending mode is additive as specified by:

$$I_{grain}[\ c\ ][\ x\ ][\ y\ ] = \text{Clip3}(\ 0, (\ 1 \ll \text{filmGrainBitDepth}[\ c\ ]\ ) - 1, I_{decoded}[\ c\ ][\ x\ ][\ y\ ] + G[\ c\ ][\ x\ ][\ y\ ]\ ) \quad \text{(D-5)}$$

– Otherwise (blending_mode_id is equal to 1), the blending mode is multiplicative as specified by:

$$\begin{aligned} I_{grain}[\ c\ ][\ x\ ][\ y\ ] = \text{Clip3}(\ 0, (\ 1 \ll \text{filmGrainBitDepth}[\ c\ ]\ ) - 1, I_{decoded}[\ c\ ][\ x\ ][\ y\ ] + \\ \text{Round}(\ (\ I_{decoded}[\ c\ ][\ x\ ][\ y\ ] * G[\ c\ ][\ x\ ][\ y\ ]\ ) \div (\ (\ 1 \ll \text{bitDepth}[\ c\ ]\ ) - 1\ )\ )\ ) \end{aligned} \quad \text{(D-6)}$$

where $I_{decoded}[\ c\ ][\ x\ ][\ y\ ]$ represents the sample value at coordinates x, y of the colour component c of the decoded image $I_{decoded}$, G[c][x][y] is the simulated film grain value at the same position and colour component, and filmGrainBitDepth[c] is the number of bits used for each sample in a fixed-length unsigned binary representation of the arrays $I_{grain}[\ c\ ][\ x\ ][\ y\ ]$, $I_{decoded}[\ c\ ][\ x\ ][\ y\ ]$, and G[c][x][y], where c = 0..2, x = 0..pic_width_in_luma_samples − 1, and y = 0..pic_height_in_luma_samples − 1.

log2_scale_factor specifies a scale factor used in the film grain characterization equations.

comp_model_present_flag[c] equal to 0 indicates that film grain is not modelled on the c-th colour component, where c equal to 0 refers to the luma component, c equal to 1 refers to the Cb component, and c equal to 2 refers to the Cr component. comp_model_present_flag[c] equal to 1 indicates that syntax elements specifying modelling of film grain on colour component c are present in the SEI message.

When separate_colour_description_present_flag is equal to 0 and chroma_format_idc is equal to 0, the value of comp_model_present_flag[1] and comp_model_present_flag[2] shall be equal to 0.

num_intensity_intervals_minus1[c] plus 1 specifies the number of intensity intervals for which a specific set of model values has been estimated.

NOTE 6 – The intensity intervals could overlap in order to simulate multi-generational film grain.

num_model_values_minus1[c] plus 1 specifies the number of model values present for each intensity interval in which the film grain has been modelled. The value of num_model_values_minus1[c] shall be in the range of 0 to 5, inclusive.

intensity_interval_lower_bound[c][i] specifies the lower bound of the i-th intensity interval for which the set of model values applies.

intensity_interval_upper_bound[c][i] specifies the upper bound of the i-th intensity interval for which the set of model values applies.

The variable intensityIntervalIdx[c][x][y][j] represents the j-th index to the list of intensity intervals selected for the sample value $I_{decoded}$[c][x][y] for c = 0..2, x = 0..pic_width_in_luma_samples − 1, y = 0..pic_height_in_luma_samples − 1, and j = 0..numApplicableIntensityIntervals[c][x][y] − 1, where numApplicableIntensityIntervals[c][x][y] is derived below.

Depending on the value of film_grain_model_id, the selection of one or more intensity intervals for the sample value $I_{decoded}$[c][x][y] is specified as follows:

– The variable numApplicableIntensityIntervals[c][x][y] is initially set equal to 0.

– If film_grain_model_id is equal to 0, the following applies:

  – The top-left sample location (xB, yB) of the current 8x8 block b that contains the sample value $I_{decoded}$[c][x][y] is derived as (xB, yB) = (x / 8, y / 8).

  – The average value b_{avg} of the current 8x8 block b is derived as follows:

```
sum8x8 = 0
for( i = 0; i < 8; i++ )
 for( j = 0; j < 8; j++ )
   sum8x8 += I_decoded[ c ][ xB * 8 + i ][ yB * 8 + j ]                      (D-7)
b_avg = Clip3( 0, 255,
( sum8x8 + ( 1 << ( filmGrainBitDepth[ c ] − 3 ) ) ) >> ( filmGrainBitDepth[ c ] − 2 ) )
```

  – The values of intensityIntervalIdx[c][x][y][j] and numApplicableIntensityIntervals[c][x][y] are derived as follows:

```
for( i = 0, j = 0; i <= num_intensity_intervals_minus1[ c ]; i++ )
 if( b_avg >= intensity_interval_lower_bound[ c ][ i ]
          && b_avg <= intensity_interval_upper_bound[ c ][ i ] ) {
     intensityIntervalIdx[ c ][ x ][ y ][ j ] = i                             (D-8)
     j++
 }
numApplicableIntensityIntervals[ c ][ x ][ y ] = j
```

– Otherwise (film_grain_model_id is equal to 1), the values of intensityIntervalIdx[c][x][y][j] and numApplicableIntensityIntervals[c][x][y] are derived as follows:

```
I_8[ c ][ x ][ y ] = ( filmGrainBitDepth[ c ] = = 8 ) ? ( I_decoded[ c ][ x ][ y ] :
        Clip3( 0, 255, ( I_decoded[ c ][ x ][ y ] +
             ( 1 << ( filmGrainBitDepth[ c ] − 9 ) ) ) >> ( filmGrainBitDepth[ c ] − 8 ) )
for( i = 0, j = 0; i <= num_intensity_intervals_minus1[ c ]; i++ )
  if( I_8[ c ][ x ][ y ] >= intensity_interval_lower_bound[ c ][ i ] &&
             I_8[ c ][ x ][ y ] <= intensity_interval_upper_bound[ c ][ i ] ) {       (D-9)
       intensityIntervalIdx[ c ][ x ][ y ][ j ] = i
       j++
```

```
    }
  numApplicableIntensityIntervals[ c ][ x ][ y ] = j
```

Samples that do not fall into any of the defined intervals (i.e., those samples for which the value of numApplicableIntensityIntervals[c][x][y] is equal to 0) are not modified by the grain generation function. Samples that fall into more than one interval (i.e., those samples for which the value of numApplicableIntensityIntervals[c][x][y] is greater than 1) will originate multi-generation grain. Multi-generation grain results from adding the grain computed independently for each of the applicable intensity intervals.

In the equations in the remainder of this clause, the variable s_j in each instance of the list comp_model_value[c][s_j] is the value of intensityIntervalIdx[c][x][y][j] derived for the sample value $I_{decoded}$[c][x][y].

comp_model_value[c][i][j] specifies the j-th model value present for the colour component c and the i-th intensity interval. The set of model values has different meaning depending on the value of film_grain_model_id.

The value of comp_model_value[c][i][j] is constrained as follows, and could be additionally constrained as specified elsewhere in this clause:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][j] shall be in the range of 0 to $2^{\text{filmGrainBitDepth[c]}} - 1$, inclusive.

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][j] shall be in the range of $-2^{(\text{ filmGrainBitDepth[c]} - 1)}$ to $2^{(\text{ filmGrainBitDepth[c]} - 1)} - 1$, inclusive.

Depending on the value of film_grain_model_id, the synthesis of the film grain is modelled as follows:

– If film_grain_model_id is equal to 0, a frequency filtering model enables simulating the original film grain for c = 0..2, x = 0..pic_width_in_luma_samples − 1 and y = 0..pic_height_in_luma_samples − 1 as specified by:

$$G[\,c\,][\,x\,][\,y\,] = (\,\text{comp\_model\_value}[\,c\,][\,s_j\,][\,0\,] * Q[\,c\,][\,x\,][\,y\,] + \text{comp\_model\_value}[\,c\,][\,s_j\,][\,5\,] * G[\,c-1\,][\,x\,][\,y\,]\,) >> \text{log2\_scale\_factor} \tag{D-10}$$

where Q[c] is a two-dimensional random process generated by filtering 16x16 blocks gaussRv with random variable elements gaussRv_{ij} generated with a normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance) and the value of an element G[c − 1][x][y] used in the right-hand side of the equation is inferred to be equal to 0 when c − 1 is less than 0.

NOTE 7 – A normalized Gaussian random variable can be generated from two independent, uniformly distributed random variables over the interval from 0 to 1 (and not equal to 0), denoted as uRv_0 and uRv_1, using the Box-Muller transformation specified by:

$$\text{gaussRv}_{i,j} = \text{Sqrt}(\,-2 * \text{Ln}(\,\text{uRv}_0\,)\,) * \text{Cos}(\,2 * \pi * \text{uRv}_1\,) \tag{D-11}$$

where π is Archimedes' constant 3.141 592 653 589 793....

The band-pass filtering of blocks gaussRv can be performed in the discrete cosine transform (DCT) domain as follows:

```
for( y = 0; y < 16; y++ )
   for( x = 0; x < 16; x++ )
      if( ( x < comp_model_value[ c ][ sj ][ 3 ]  &&  y < comp_model_value[ c ][ sj ][ 4 ] ) | |    (D-12)
         x > comp_model_value[ c ][ sj ][ 1 ]  | |  y > comp_model_value[ c ][ sj ][ 2 ] )
         gaussRv[ x ][ y ] = 0
filteredRv = IDCT16x16( gaussRv )
```

where IDCT16x16(z) refers to a unitary inverse discrete cosine transformation (IDCT) operating on a 16x16 matrix argument z as specified by:

$$\text{IDCT16x16}(\,z\,) = r * z * r^T \tag{D-13}$$

where the superscript T indicates a matrix transposition and r is the 16x16 matrix with elements r_{ij} specified by:

$$r_{i,j} = \frac{(\,(\,i == 0\,)\,?\,1 : \text{Sqrt}(\,2\,)\,)}{4} * \text{Cos}\left(\frac{i * (\,2 * j + 1\,) * \pi}{32}\right) \tag{D-14}$$

where π is Archimedes' constant 3.141 592 653 589 793....

Q[c] is formed by the frequency-filtered blocks filteredRv.

NOTE 8 – Coded model values are based on blocks of size 16x16, but a decoder implementation could use other block sizes. For example, decoders implementing the IDCT on 8x8 blocks could down-convert by a factor of two the set of coded model values comp_model_value[c][s_j][i] for i equal to 1..4.

NOTE 9 – To reduce the degree of visible blocks that can result from mosaicking the frequency-filtered blocks filteredRv, decoders could apply a low-pass filter to the boundaries between frequency-filtered blocks.

– Otherwise (film_grain_model_id is equal to 1), an auto-regression model enables simulating the original film grain for c = 0..2, x = 0..pic_width_in_luma_samples − 1, and y = 0..pic_height_in_luma_samples − 1 as specified by:

$$\begin{aligned}
&G[\,c\,][\,x\,][\,y\,] = (\,\text{comp_model_value}[\,c\,][\,s_j\,][\,0\,] * n[\,c\,][\,x\,][\,y\,] + \\
&\quad \text{comp_model_value}[\,c\,][\,s_j\,][\,1\,] * (\,G[\,c\,][\,x-1\,][\,y\,] + (\,(\,\text{comp_model_value}[\,c\,][\,s_j\,][\,4\,] * \\
&\quad G[\,c\,][\,x\,][\,y-1\,]\,) >> \\
&\qquad \text{log2_scale_factor}\,)\,) + \\
&\quad \text{comp_model_value}[\,c\,][\,s_j\,][\,3\,] * (\,(\,(\,\text{comp_model_value}[\,c\,][\,s_j\,][\,4\,] * G[\,c\,][\,x-1\,][\,y-1\,]\,) \\
&\quad >> \\
&\qquad \text{log2_scale_factor}\,) + G[\,c\,][\,x+1\,][\,y-1\,]\,) + \\
&\quad \text{comp_model_value}[\,c\,][\,s_j\,][\,5\,] * (\,G[\,c\,][\,x-2\,][\,y\,] + \\
&\qquad (\,(\,\text{comp_model_value}[\,c\,][\,s_j\,][\,4\,] * \text{comp_model_value}[\,c\,][\,s_j\,][\,4\,] * G[\,c\,][\,x\,][\,y-2\,]\,) >> \\
&\qquad\quad (\,2 * \text{log2_scale_factor}\,)\,)\,) + \\
&\quad \text{comp_model_value}[\,c\,][\,s_j\,][\,2\,] * G[\,c-1\,][\,x\,][\,y\,]\,) >> \text{log2_scale_factor}
\end{aligned} \tag{D-15}$$

where n[c][x][y] is a random variable with normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance for each value of c, x, and y) and the value of an element G[c][x][y] used in the right-hand side of the equation is inferred to be equal to 0 when any of the following conditions are true:

– x is less than 0,

– y is less than 0,

– c is less than 0.

comp_model_value[c][i][0] provides the first model value for the model as specified by film_grain_model_id. comp_model_value[c][i][0] corresponds to the standard deviation of the Gaussian noise term in the generation functions specified in Equations D-10 through D-15.

comp_model_value[c][i][1] provides the second model value for the model as specified by film_grain_model_id. When film_grain_model_id is equal to 0, comp_model_value[c][i][1] shall be greater than or equal to 0 and less than 16.

When not present in the film grain characteristics SEI message, comp_model_value[c][i][1] is inferred as follows:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][1] is inferred to be equal to 8.

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][1] is inferred to be equal to 0.

comp_model_value[c][i][1] is interpreted as follows:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][1] indicates the horizontal high cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][1] indicates the first order spatial correlation for neighbouring samples at positions (x − 1, y) and (x, y − 1).

comp_model_value[c][i][2] provides the third model value for the model as specified by film_grain_model_id. When film_grain_model_id is equal to 0, comp_model_value[c][i][2] shall be greater than or equal to 0 and less than 16.

When not present in the film grain characteristics SEI message, comp_model_value[c][i][2] is inferred as follows:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][2] is inferred to be equal to comp_model_value[c][i][1]

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][2] is inferred to be equal to 0.

comp_model_value[c][i][2] is interpreted as follows:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][2] indicates the vertical high cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][2] indicates the colour correlation between consecutive colour components.

comp_model_value[c][i][3] provides the fourth model value for the model as specified by film_grain_model_id. When film_grain_model_id is equal to 0, comp_model_value[c][i][3] shall be greater than or equal to 0 and less than or equal to comp_model_value[c][i][1].

When not present in the film grain characteristics SEI message, comp_model_value[c][i][3] is inferred to be equal to 0.

comp_model_value[c][i][3] is interpreted as follows:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][3] indicates the horizontal low cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][3] indicates the first order spatial correlation for neighbouring samples at positions (x − 1, y − 1) and (x + 1, y − 1).

comp_model_value[c][i][4] provides the fifth model value for the model as specified by film_grain_model_id. When film_grain_model_id is equal to 0, comp_model_value[c][i][4] shall be greater than or equal to 0 and less than or equal to comp_model_value[c][i][2].

When not present in the film grain characteristics SEI message, comp_model_value[c][i][4] is inferred to be equal to film_grain_model_id.

comp_model_value[c][i][4] is interpreted as follows:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][4] indicates the vertical low cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][4] indicates the aspect ratio of the modelled grain.

comp_model_value[c][i][5] provides the sixth model value for the model as specified by film_grain_model_id.

When not present in the film grain characteristics SEI message, comp_model_value[c][i][5] is inferred to be equal to 0.

comp_model_value[c][i][5] is interpreted as follows:

– If film_grain_model_id is equal to 0, comp_model_value[c][i][5] indicates the colour correlation between consecutive colour components.

– Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][5] indicates the second order spatial correlation for neighbouring samples at positions (x, y − 2) and (x − 2, y).

film_grain_characteristics_persistence_flag specifies the persistence of the film grain characteristics SEI message for the current layer.

film_grain_characteristics_persistence_flag equal to 0 specifies that the film grain characteristics SEI message applies to the current decoded picture only.

Let picA be the current picture. film_grain_characteristics_persistence_flag equal to 1 specifies that the film grain characteristics SEI message persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a film grain characteristics SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

D.3.14 Post-filter hint SEI message semantics

This SEI message provides the coefficients of a post-filter or correlation information for the design of a post-filter for potential use in post-processing of the current picture after it is decoded and output to obtain improved displayed quality.

filter_hint_size_y specifies the vertical size of the filter coefficient or correlation array. The value of filter_hint_size_y shall be in the range of 1 to 15, inclusive.

filter_hint_size_x specifies the horizontal size of the filter coefficient or correlation array. The value of filter_hint_size_x shall be in the range of 1 to 15, inclusive.

filter_hint_type identifies the type of the transmitted filter hints as specified in Table D.6. The value of filter_hint_type shall be in the range of 0 to 2, inclusive. The value of filter_hint_type equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore post-filter hint SEI messages having filter_hint_type equal to 3.

Table D.6 – filter_hint_type values

Value	Description
0	Coefficients of a 2D-FIR filter
1	Coefficients of two 1D-FIR filters
2	Cross-correlation matrix

filter_hint_value[cIdx][cy][cx] specifies a filter coefficient or an element of a cross-correlation matrix between the original and the decoded signal with 16-bit precision. The value of filter_hint_value[cIdx][cy][cx] shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive. cIdx specifies the related colour component, cy represents a counter in vertical direction and cx represents a counter in horizontal direction. Depending on the value of filter_hint_type, the following applies:

– If filter_hint_type is equal to 0, the coefficients of a 2-dimensional finite impulse response (FIR) filter with the size of filter_hint_size_y * filter_hint_size_x are transmitted.

– Otherwise, if filter_hint_type is equal to 1, the filter coefficients of two 1-dimensional FIR filters are transmitted. In this case, filter_hint_size_y shall be equal to 2. The index cy equal to 0 specifies the filter coefficients of the horizontal filter and cy equal to 1 specifies the filter coefficients of the vertical filter. In the filtering process, the horizontal filter is applied first and the result is filtered by the vertical filter.

– Otherwise (filter_hint_type is equal to 2), the transmitted hints specify a cross-correlation matrix between the original signal s and the decoded signal s′.

NOTE 1 – The normalized cross-correlation matrix for a related colour component identified by cIdx with the size of filter_hint_size_y * filter_hint_size_x is defined as follows:

$$\text{filter\_hint\_value}(\text{cIdx}, \text{cy}, \text{cx}) = \frac{1}{(2^{8+\text{bitDepth}} - 1)^2 * \text{h} * \text{w}} \sum_{m=0}^{h-1} \sum_{n=0}^{w-1} s(m, n) * s'(m + \text{cy} - \text{OffsetY}, n + \text{cx} - \text{OffsetX}) \qquad \text{(D-16)}$$

where s denotes array of samples of the colour component cIdx of the original picture, s′ denotes corresponding array of the decoded picture, h denotes the vertical height of the related colour component, w denotes the horizontal width of the related colour component, bitDepth denotes the bit depth of the colour component, OffsetY is equal to (filter_hint_size_y >> 1), OffsetX is equal to (filter_hint_size_x >> 1), 0 <= cy < filter_hint_size_y and 0 <= cx < filter_hint_size_x.

NOTE 2 – A decoder can derive a Wiener post-filter from the cross-correlation matrix of original and decoded signal and the auto-correlation matrix of the decoded signal.

D.3.15 Tone mapping information SEI message semantics

This SEI message provides information to enable remapping of the colour samples of the output decoded pictures for customization to particular display environments. The remapping process maps coded sample values in the RGB colour space (specified in Annex E) to target sample values. The mappings are expressed either in the luma or RGB colour space domain and should be applied to the luma component or to each RGB component produced by colour space conversion of the decoded image accordingly.

tone_map_id contains an identifying number that may be used to identify the purpose of the tone mapping model. The value of tone_map_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of tone_map_id from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of tone_map_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of tone_map_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 1 – The tone_map_id can be used to support tone mapping operations that are suitable for different display scenarios. For example, different values of tone_map_id may correspond to different display bit depths.

tone_map_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous tone mapping information SEI message in output order that applies to the current layer. tone_map_cancel_flag equal to 0 indicates that tone mapping information follows.

tone_map_persistence_flag specifies the persistence of the tone mapping information SEI message.

tone_map_persistence_flag equal to 0 specifies that the tone mapping information applies to the current decoded picture only.

Let picA be the current picture. tone_map_persistence_flag equal to 1 specifies that the tone mapping information persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.

– A picture picB in the current layer in an access unit containing a tone mapping information SEI message with the same value of tone_map_id and applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

coded_data_bit_depth specifies the BitDepth_Y for interpretation of the luma component of the associated pictures for purposes of interpretation of the tone mapping information SEI message. When tone mapping information SEI messages are present that have coded_data_bit_depth that is not equal to BitDepth_Y, these refer to the hypothetical result of a transcoding operation performed to convert the coded video to the BitDepth_Y corresponding to the value of coded_data_bit_depth.

The value of coded_data_bit_depth shall be in the range of 8 to 14, inclusive. Values of coded_data_bit_depth from 0 to 7 and from 15 to 255 are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all tone mapping SEI messages that contain a coded_data_bit_depth in the range of 0 to 7, inclusive, or in the range of 15 to 255, inclusive, and bitstreams shall not contain such values.

target_bit_depth specifies the bit depth of the output of the dynamic range mapping function (or tone mapping function) described by the tone mapping information SEI message. The tone mapping function specified with a particular target_bit_depth is suggested to be reasonable for all display bit depths that are less than or equal to the target_bit_depth.

The value of target_bit_depth shall be in the range of 1 to 16, inclusive. Values of target_bit_depth equal to 0 and in the range of 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all tone mapping SEI messages that contain a value of target_bit_depth equal to 0 or in the range of 17 to 255, inclusive, and bitstreams shall not contain such values.

tone_map_model_id specifies the model utilized for mapping the coded data into the target_bit_depth range. Values greater than 4 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore all tone mapping SEI messages that contain a value of tone_map_model_id greater than 4 and bitstreams shall not contain such values.

NOTE 2 – A tone_map_model_id of 0 corresponds to a linear mapping with clipping; a tone_map_model_id of 1 corresponds to a sigmoidal mapping; a tone_map_model_id of 2 corresponds to a user-defined table mapping, and a tone_map_model_id of 3 corresponds to a piece-wise linear mapping, tone_map_model_id of 4 corresponds to luminance dynamic range information.

min_value specifies the RGB sample value that maps to the minimum value in the bit depth indicated by target_bit_depth. It is used in combination with the max_value parameter. All sample values in the decoded picture that are less than or equal to min_value, after conversion to RGB as necessary, are mapped to this minimum value in the target_bit_depth representation.

max_value specifies the RGB sample value that maps to the maximum value in the bit depth indicated by target_bit_depth. It is used in combination with the min_value parameter. All sample values in the decoded picture that are greater than or equal to max_value, after conversion to RGB as necessary, are mapped to this maximum value in the target_bit_depth representation.

When present, max_value shall be greater than or equal to min_value.

sigmoid_midpoint specifies the RGB sample value of the coded data that are mapped to the centre point of the target_bit_depth representation. It is used in combination with the sigmoid_width parameter.

sigmoid_width specifies the distance between two coded data values that approximately correspond to the 5% and 95% values of the target_bit_depth representation, respectively. It is used in combination with the sigmoid_midpoint parameter and is interpreted according to the following function:

$$f(i) = \text{Round}\left(\frac{2^{target\_bit\_depth} - 1}{1 + \exp\left(\frac{-6 * (i - sigmoid\_midpoint)}{sigmoid\_width} \right)} \right) \quad \text{(D-17)}$$

where f(i) denotes the function that maps an RGB sample value i from the coded data to a resulting RGB sample value in the target_bit_depth representation.

start_of_coded_interval[i] specifies the beginning point of an interval in the coded data such that all RGB sample values that are greater than or equal to start_of_coded_interval[i] and less than start_of_coded_interval[i + 1] are mapped to i

in the target bit depth representation. The value of start_of_coded_interval[$2^{target\_bit\_depth}$] is equal to $2^{coded\_data\_bit\_depth}$. The number of bits used for the representation of the start_of_coded_interval is ((coded_data_bit_depth + 7) >> 3) << 3.

num_pivots specifies the number of pivot points in the piece-wise linear mapping function without counting the two default end points, (0, 0) and ($2^{coded\_data\_bit\_depth} - 1$, $2^{target\_bit\_depth} - 1$).

coded_pivot_value[i] specifies the value in the coded_data_bit_depth corresponding to the i-th pivot point. The number of bits used for the representation of the coded_pivot_value is ((coded_data_bit_depth + 7) >> 3) << 3.

target_pivot_value[i] specifies the value in the reference target_bit_depth corresponding to the i-th pivot point. The number of bits used for the representation of the target_pivot_value is ((target_bit_depth + 7) >> 3) << 3.

camera_iso_speed_idc indicates the camera ISO speed for daylight illumination as specified in ISO 12232, interpreted as specified in Table D.7. When camera_iso_speed_idc indicates EXTENDED_ISO, the ISO speed is indicated by camera_iso_speed_value.

camera_iso_speed_value indicates the camera ISO speed for daylight illumination as specified in ISO 12232 when camera_iso_speed_idc indicates EXTENDED_ISO. The value of camera_iso_speed_value shall not be equal to 0.

exposure_idx_idc indicates the exposure index setting of the camera as specified in ISO 12232, interpreted as specified in Table D.7. When exposure_idx_idc indicates EXTENDED_ISO, the exposure index is indicated by exposure_idx_value.

The values of camera_iso_speed_idc and exposure_idx_idc in the range of 31 to 254, inclusive, are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore tone mapping SEI messages that contain these values.

exposure_idx_value indicates the exposure index setting of the camera as specified in ISO 12232 when exposure_idx_idc indicates EXTENDED_ISO. The value of exposure_idx_value shall not be equal to 0.

Table D.7 – Interpretation of camera_iso_speed_idc and exposure_idx_idc

camera_iso_speed_idc or exposure_idx_idc	Indicated value
0	Unspecified
1	10
2	12
3	16
4	20
5	25
6	32
7	40
8	50
9	64
10	80
11	100
12	125
13	160
14	200
15	250
16	320
17	400
18	500
19	640
20	800
21	1 000
22	1 250
23	1 600
24	2 000
25	2 500
26	3 200
27	4 000
28	5 000
29	6 400
30	8 000
31..254	Reserved
255	EXTENDED_ISO

exposure_compensation_value_sign_flag, when applicable as specified below, specifies the sign of the variable ExposureCompensationValue that indicates the exposure compensation value setting used for the process of image production.

exposure_compensation_value_numerator, when applicable as specified below, specifies the numerator of the variable ExposureCompensationValue that indicates the exposure compensation value setting used for the process of image production.

exposure_compensation_value_denom_idc, when not equal to 0, specifies the denominator of the variable ExposureCompensationValue that indicates the exposure compensation value setting used for the process of image production.

When exposure_compensation_value_denom_idc is present and not equal to 0, the variable ExposureCompensationValue is derived from exposure_compensation_value_sign_flag, exposure_compensation_value_numerator and exposure_compensation_value_denom_idc. exposure_compensation_value_sign_flag equal to 0 indicates that the ExposureCompensationValue is positive. exposure_compensation_value_sign_flag equal to 1 indicates that the ExposureCompensationValue is negative. When ExposureCompensationValue is positive, the image is indicated to have been further sensitized through the process of production, relative to the recommended exposure index of the camera as specified in ISO 12232. When ExposureCompensationValue is negative, the image is indicated to have been further desensitized through the process of production, relative to the recommended exposure index of the camera as specified in ISO 12232.

When exposure_compensation_value_denom_idc is present and not equal to 0, the variable ExposureCompensationValue is derived as follows:

$$\text{ExposureCompensationValue} = (1 - 2 * \text{exposure\_compensation\_value\_sign\_flag}) * \\ \text{exposure\_compensation\_value\_numerator} \div \\ \text{exposure\_compensation\_value\_denom\_idc} \quad \text{(D-18)}$$

The value of ExposureCompensationValue is interpreted in units of exposure steps such that an increase of 1 in ExposureCompensationValue corresponds to a doubling of exposure in units of lux-seconds. For example, the exposure compensation value equal to +1÷2 at the production stage may be indicated by setting exposure_compensation_value_sign_flag to 0, exposure_compensation_value_numerator to 1 and exposure_compensation_value_denom_idc to 2.

When exposure_compensation_value_denom_idc is present and equal to 0, the exposure compensation value is indicated as unknown or unspecified.

ref_screen_luminance_white indicates the reference screen brightness setting for the extended white level used for image production process in units of candelas per square metre.

extended_range_white_level indicates the luminance dynamic range for extended dynamic-range display of the associated pictures, after conversion to the linear light domain for display, expressed as an integer percentage relative to the nominal white level. The value of extended_range_white_level should be greater than or equal to 100.

nominal_black_level_code_value indicates the luma sample value of the associated decoded pictures to which the nominal black level is assigned. For example, when coded_data_bit_depth is equal to 8, video_full_range_flag is equal to 0, and matrix_coeffs is equal to 1, nominal_black_level_code_value should be equal to 16.

nominal_white_level_code_value indicates the luma sample value of the associated decoded pictures to which the nominal white level is assigned. For example, when coded_data_bit_depth is equal to 8, video_full_range_flag is equal to 0 and matrix_coeffs is equal to 1, nominal_white_level_code_value should be equal to 235. When present, the value of nominal_white_level_code_value shall be greater than nominal_black_level_code_value.

extended_white_level_code_value indicates the luma sample value of the associated decoded pictures to which the white level associated with an extended dynamic range is assigned. When present, the value of extended_white_level_code_value shall be greater than or equal to nominal_white_level_code_value.

D.3.16 Frame packing arrangement SEI message semantics

This SEI message informs the decoder that the output cropped decoded picture contains samples of multiple distinct spatially packed constituent frames that are packed into one frame, or that the output cropped decoded pictures in output order form a temporal interleaving of alternating first and second constituent frames, using an indicated frame packing arrangement scheme. This information can be used by the decoder to appropriately rearrange the samples and process the samples of the constituent frames appropriately for display or other purposes (which are outside the scope of this Specification).

This SEI message may be associated with pictures that are either frames (when field_seq_flag is equal to 0) or fields (when field_seq_flag is equal to 1). The frame packing arrangement of the samples is specified in terms of the sampling structure of a frame in order to define a frame packing arrangement structure that is invariant with respect to whether a picture is a single field of such a packed frame or is a complete packed frame.

When general_non_packed_constraint_flag is equal to 1 for a CVS, there shall be no frame packing arrangement SEI messages in the CVS.

frame_packing_arrangement_id contains an identifying number that may be used to identify the usage of the frame packing arrangement SEI message. The value of frame_packing_arrangement_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of frame_packing_arrangement_id from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of frame_packing_arrangement_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of frame_packing_arrangement_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

frame_packing_arrangement_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous frame packing arrangement SEI message in output order that applies to the current layer. frame_packing_arrangement_cancel_flag equal to 0 indicates that frame packing arrangement information follows.

frame_packing_arrangement_type identifies the indicated interpretation of the sample arrays of the output cropped decoded picture as specified in Table D.8.

When frame_packing_arrangement_type is equal to 3 or 4, each component plane of the output cropped decoded picture contains all samples (when field_pic_flag is equal to 0) or the samples corresponding to the top or bottom field (when field_pic_flag is equal to 1) of the samples of a frame packing arrangement structure.

Table D.8 – Definition of frame_packing_arrangement_type

Value	Interpretation
3	The frame packing arrangement structure contains a side-by-side packing arrangement of corresponding planes of two constituent frames as illustrated in Figure D.4, Figure D.5 and Figure D.8.
4	The frame packing arrangement structure contains a top-bottom packing arrangement of corresponding planes of two constituent frames as illustrated in Figure D.6 and Figure D.7.
5	The component planes of the output cropped decoded pictures in output order form a temporal interleaving of alternating first and second constituent frames as illustrated in Figure D.9.

NOTE 1 – Figure D.4 to Figure D.8 provide typical examples of rearrangement and upconversion processing for various packing arrangement schemes. Actual characteristics of the constituent frames are signalled in detail by the subsequent syntax elements of the frame packing arrangement SEI message. In Figure D.4 to Figure D.8, an upconversion processing is performed on each constituent frame to produce frames having the same resolution as that of the decoded frame. An example of the upsampling method to be applied to a quincunx sampled frame as shown in Figure D.8 is to fill in missing positions with an average of the available spatially neighbouring samples (the average of the values of the available samples above, below, to the left and to the right of each sample to be generated). The actual upconversion process to be performed, if any, is outside the scope of this Specification.

NOTE 2 – When the output time of the samples of constituent frame 0 differs from the output time of the samples of constituent frame 1 (i.e., when field_views_flag is equal to 1 or frame_packing_arrangement_type is equal to 5) and the display system in use presents two views simultaneously, the display time for constituent frame 0 should be delayed to coincide with the display time for constituent frame 1. (The display process is not specified in this Specification.)

NOTE 3 – When field_views_flag is equal to 1 or frame_packing_arrangement_type is equal to 5, the value 0 for fixed_pic_rate_within_cvs_flag is not expected to be prevalent in industry use of this SEI message.

NOTE 4 – frame_packing_arrangement_type equal to 5 describes a temporal interleaving process of different views.

All other values of frame_packing_arrangement_type are reserved for future use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that bitstreams conforming to this version of this Specification shall not contain such other values of frame_packing_arrangement_type. Decoders shall ignore frame packing arrangement SEI messages that contain reserved values of frame_packing_arrangement_type.

quincunx_sampling_flag equal to 1 indicates that each colour component plane of each constituent frame is quincunx sampled as illustrated in Figure D.8 and quincunx_sampling_flag equal to 0 indicates that the colour component planes of each constituent frame are not quincunx sampled.

When frame_packing_arrangement_type is equal to 5, it is a requirement of bitstream conformance that quincunx_sampling_flag shall be equal to 0.

NOTE 5 – For any chroma format (4:2:0, 4:2:2 or 4:4:4), the luma plane and each chroma plane is quincunx sampled as illustrated in Figure D.8 when quincunx_sampling_flag is equal to 1.

content_interpretation_type indicates the intended interpretation of the constituent frames as specified in Table D.9. Values of content_interpretation_type that do not appear in Table D.9 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore frame packing arrangement SEI messages that contain reserved values of content_interpretation_type.

For each specified frame packing arrangement scheme, there are two constituent frames that are referred to as frame 0 and frame 1.

Table D.9 – Definition of content_interpretation_type

Value	Interpretation
0	Unspecified relationship between the frame packed constituent frames
1	Indicates that the two constituent frames form the left and right views of a stereo view scene, with frame 0 being associated with the left view and frame 1 being associated with the right view
2	Indicates that the two constituent frames form the right and left views of a stereo view scene, with frame 0 being associated with the right view and frame 1 being associated with the left view

NOTE 6 – The value 2 for content_interpretation_type is not expected to be prevalent in industry use of this SEI message. However, the value was specified herein for purposes of completeness.

spatial_flipping_flag equal to 1, when frame_packing_arrangement_type is equal to 3 or 4, indicates that one of the two constituent frames is spatially flipped relative to its intended orientation for display or other such purposes.

When frame_packing_arrangement_type is equal to 3 or 4 and spatial_flipping_flag is equal to 1, the type of spatial flipping that is indicated is as follows:

– If frame_packing_arrangement_type is equal to 3, the indicated spatial flipping is horizontal flipping.

– Otherwise (frame_packing_arrangement_type is equal to 4), the indicated spatial flipping is vertical flipping.

When frame_packing_arrangement_type is not equal to 3 or 4, it is a requirement of bitstream conformance that spatial_flipping_flag shall be equal to 0. When frame_packing_arrangement_type is not equal to 3 or 4, the value 1 for spatial_flipping_flag is reserved for future use by ITU-T | ISO/IEC. When frame_packing_arrangement_type is not equal to 3 or 4, decoders shall ignore the value 1 for spatial_flipping_flag.

frame0_flipped_flag, when spatial_flipping_flag is equal to 1, indicates which one of the two constituent frames is flipped.

When spatial_flipping_flag is equal to 1, frame0_flipped_flag equal to 0 indicates that frame 0 is not spatially flipped and frame 1 is spatially flipped and frame0_flipped_flag equal to 1 indicates that frame 0 is spatially flipped and frame 1 is not spatially flipped.

When spatial_flipping_flag is equal to 0, it is a requirement of bitstream conformance that frame0_flipped_flag shall be equal to 0. When spatial_flipping_flag is equal to 0, the value 1 for spatial_flipping_flag is reserved for future use by ITU-T | ISO/IEC. When spatial_flipping_flag is equal to 0, decoders shall ignore the value of frame0_flipped_flag.

field_views_flag equal to 1 indicates that all pictures in the current CVS are coded as fields, all fields of a particular parity are considered a first constituent frame and all fields of the opposite parity are considered a second constituent frame. It is a requirement of bitstream conformance that the field_views_flag shall be equal to 0, the value 1 for field_views_flag is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of field_views_flag.

current_frame_is_frame0_flag equal to 1, when frame_packing_arrangement_type is equal to 5, indicates that the current decoded frame is constituent frame 0 and the next decoded frame in output order is constituent frame 1 and the display time of the constituent frame 0 should be delayed to coincide with the display time of constituent frame 1. current_frame_is_frame0_flag equal to 0, when frame_packing_arrangement_type is equal to 5, indicates that the current decoded frame is constituent frame 1 and the previous decoded frame in output order is constituent frame 0 and the display time of the constituent frame 1 should not be delayed for purposes of stereo-view pairing.

When frame_packing_arrangement_type is not equal to 5, the constituent frame associated with the upper-left sample of the decoded frame is considered to be constituent frame 0 and the other constituent frame is considered to be constituent frame 1. When frame_packing_arrangement_type is not equal to 5, it is a requirement of bitstream conformance that current_frame_is_frame0_flag shall be equal to 0. When frame_packing_arrangement_type is not equal to 5, the value 1 for current_frame_is_frame0_flag is reserved for future use by ITU-T | ISO/IEC. When frame_packing_arrangement_type is not equal to 5, decoders shall ignore the value of current_frame_is_frame0_flag.

frame0_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the samples of constituent frame 0 of the CVS refer to samples of any constituent frame 1. frame0_self_contained_flag equal to 0 indicates that some inter prediction operations within the decoding process for the samples of constituent frame 0 of the CVS may or may not refer to samples of some constituent frame 1. Within a CVS, the value of frame0_self_contained_flag in all frame packing arrangement SEI messages shall be the same.

frame1_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the samples of constituent frame 1 of the CVS refer to samples of any constituent frame 0. frame1_self_contained_flag equal to 0 indicates that some inter prediction operations within the decoding process for the samples of constituent frame 1 of the CVS may or may not refer to samples of some constituent frame 0. Within a CVS, the value of frame1_self_contained_flag in all frame packing arrangement SEI messages shall be the same.

When quincunx_sampling_flag is equal to 0 and frame_packing_arrangement_type is not equal to 5, two (x, y) coordinate pairs are specified to determine the indicated luma sampling grid alignment for constituent frame 0 and constituent frame 1, relative to the upper left corner of the rectangular area represented by the samples of the corresponding constituent frame.

NOTE 7 – The location of chroma samples relative to luma samples can be indicated by the chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field syntax elements in the video usability information (VUI) parameters.

frame0_grid_position_x (when present) specifies the x component of the (x, y) coordinate pair for constituent frame 0.

frame0_grid_position_y (when present) specifies the y component of the (x, y) coordinate pair for constituent frame 0.

frame1_grid_position_x (when present) specifies the x component of the (x, y) coordinate pair for constituent frame 1.

frame1_grid_position_y (when present) specifies the y component of the (x, y) coordinate pair for constituent frame 1.

When quincunx_sampling_flag is equal to 0 and frame_packing_arrangement_type is not equal to 5 the (x, y) coordinate pair for each constituent frame is interpreted as follows:

– If the (x, y) coordinate pair for a constituent frame is equal to (0, 0), this indicates a default sampling grid alignment specified as follows:
  – If frame_packing_arrangement_type is equal to 3, the indicated position is the same as for the (x, y) coordinate pair value (4, 8), as illustrated in Figure D.4.
  – Otherwise (frame_packing_arrangement_type is equal to 4), the indicated position is the same as for the (x, y) coordinate pair value (8, 4), as illustrated in Figure D.6.
– Otherwise, if the (x, y) coordinate pair for a constituent frame is equal to (15, 15), this indicates that the sampling grid alignment is unknown or unspecified or specified by other means not specified in this Specification.
– Otherwise, the x and y elements of the (x, y) coordinate pair specify the indicated horizontal and vertical sampling grid alignment positioning to the right of and below the upper left corner of the rectangular area represented by the corresponding constituent frame, respectively, in units of one sixteenth of the luma sample grid spacing between the samples of the columns and rows of the constituent frame that are present in the decoded frame (prior to any upsampling for display or other purposes).

NOTE 8 – The spatial location reference information frame0_grid_position_x, frame0_grid_position_y, frame1_grid_position_x, and frame1_grid_position_y is not provided when quincunx_sampling_flag is equal to 1 because the spatial alignment in this case is assumed to be such that constituent frame 0 and constituent frame 1 cover corresponding spatial areas with interleaved quincunx sampling patterns as illustrated in Figure D.8.

frame_packing_arrangement_reserved_byte is reserved for future use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that the value of frame_packing_arrangement_reserved_byte shall be equal to 0. All other values of frame_packing_arrangement_reserved_byte are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of frame_packing_arrangement_reserved_byte.

frame_packing_arrangement_persistence_flag specifies the persistence of the frame packing arrangement SEI message for the current layer.

frame_packing_arrangement_persistence_flag equal to 0 specifies that the frame packing arrangement SEI message applies to the current decoded frame only.

Let picA be the current picture. frame_packing_arrangement_persistence_flag equal to 1 specifies that the frame packing arrangement SEI message persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.
– The bitstream ends.
– A frame picB in the current layer in an access unit containing a frame packing arrangement SEI message with the same value of frame_packing_arrangement_id and applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

upsampled_aspect_ratio_flag equal to 1 indicates that the sample aspect ratio (SAR) indicated by the VUI parameters of the SPS identifies the SAR of the samples after the application of an upconversion process to produce a higher resolution

frame from each constituent frame as illustrated in Figure D.4 to Figure D.8. upsampled_aspect_ratio_flag equal to 0 indicates that the SAR indicated by the VUI parameters of the SPS identifies the SAR of the samples before the application of any such upconversion process.

NOTE 9 – The default display window parameters in the VUI parameters of the SPS can be used by an encoder to indicate to a decoder that does not interpret the frame packing arrangement SEI message that the default display window is an area within only one of the two constituent frames.

NOTE 10 – The SAR indicated in the VUI parameters should indicate the preferred display picture shape for the packed decoded frame output by a decoder that does not interpret the frame packing arrangement SEI message. When upsampled_aspect_ratio_flag is equal to 1, the SAR produced in each up-converted colour plane is indicated to be the same as the SAR indicated in the VUI parameters in the examples shown in Figure D.4 to Figure D.8. When upsampled_aspect_ratio_flag is equal to 0, the SAR produced in each colour plane prior to upconversion is indicated to be the same as the SAR indicated in the VUI parameters in the examples shown in Figure D.4 to Figure D.8.

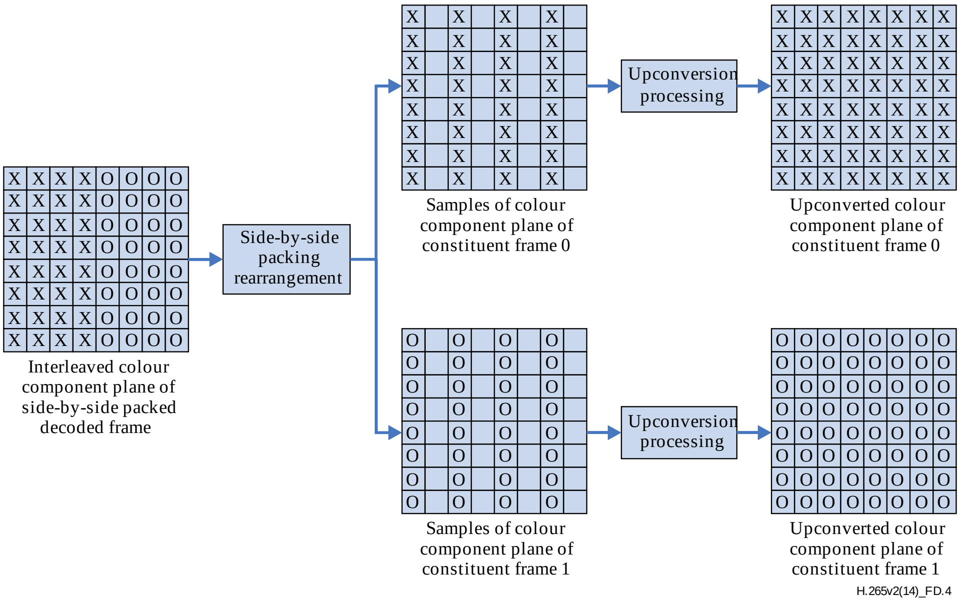


Figure D.4 – Rearrangement and upconversion of side-by-side packing arrangement with frame_packing_arrangement_type equal to 3, quincunx_sampling_flag equal to 0 and (x, y) equal to (0, 0) or (4, 8) for both constituent frames

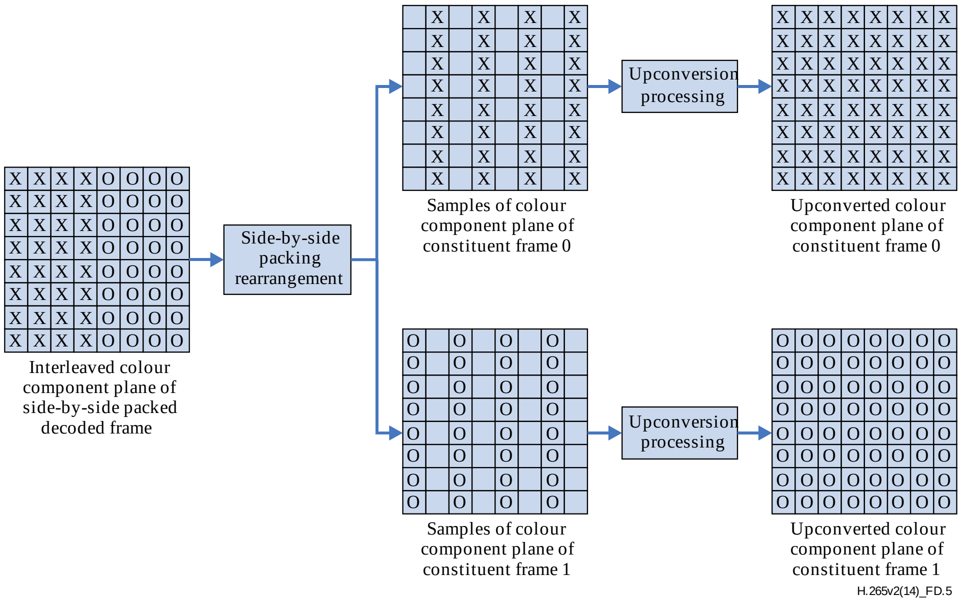


Figure D.5 – Rearrangement and upconversion of side-by-side packing arrangement with frame_packing_arrangement_type equal to 3, quincunx_sampling_flag equal to 0, (x, y) equal to (12, 8) for constituent frame 0 and (x, y) equal to (0, 0) or (4, 8) for constituent frame 1

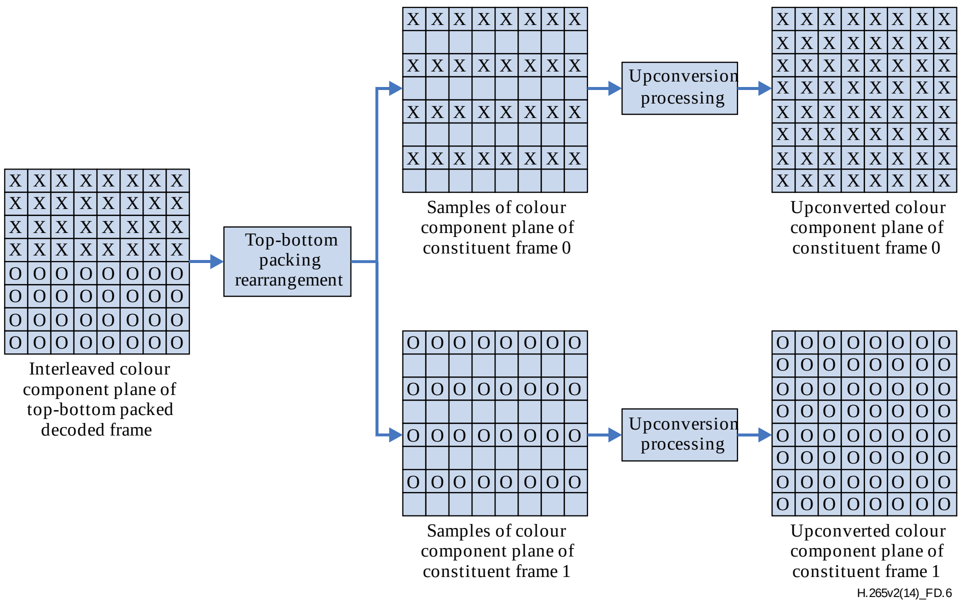


Figure D.6 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0 and (x, y) equal to (0, 0) or (8, 4) for both constituent frames

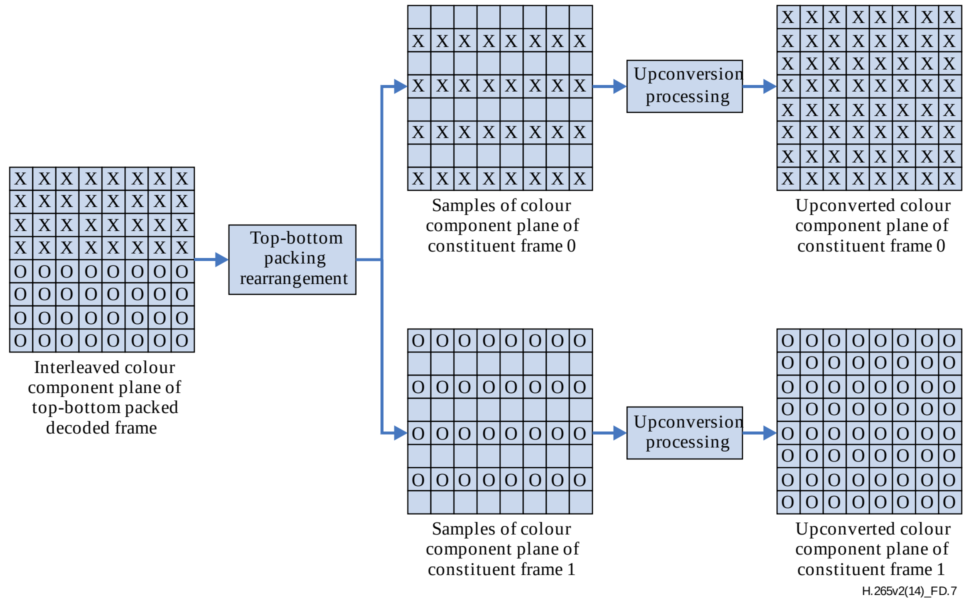


Figure D.7 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0, (x, y) equal to (8, 12) for constituent frame 0 and (x, y) equal to (0, 0) or (8, 4) for constituent frame 1

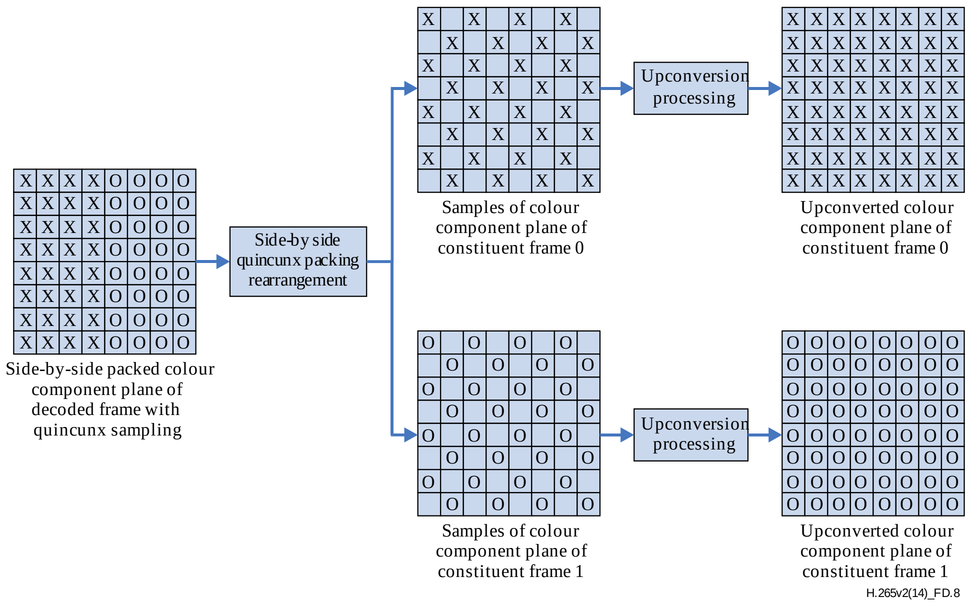


Figure D.8 – Rearrangement and upconversion of side-by-side packing arrangement with quincunx sampling (frame_packing_arrangement_type equal to 3 with quincunx_sampling_flag equal to 1)

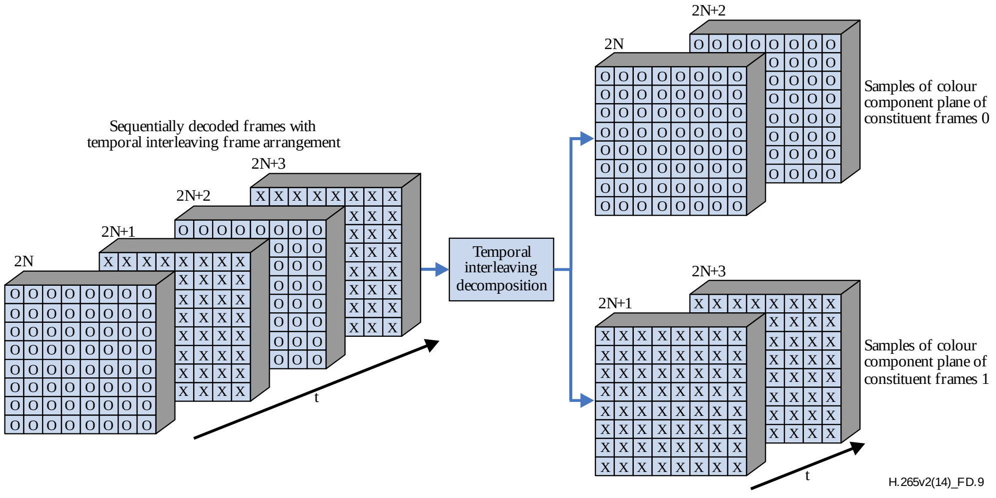


Figure D.9 – Rearrangement of a temporal interleaving frame arrangement (frame_packing_arrangement_type equal to 5)

D.3.17 Display orientation SEI message semantics

When the associated picture has PicOutputFlag equal to 1, the display orientation SEI message informs the decoder of a transformation that is recommended to be applied to the cropped decoded picture prior to display.

display_orientation_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous display orientation SEI message in output order. display_orientation_cancel_flag equal to 0 indicates that display orientation information follows.

hor_flip equal to 1 indicates that the cropped decoded picture should be flipped horizontally for display. hor_flip equal to 0 indicates that the decoded picture should not be flipped horizontally.

When hor_flip is equal to 1, the cropped decoded picture should be flipped as follows for each component Z equal to Y, Cb and Cr, letting dZ be the final cropped array of output samples for the component Z:

```
for( x = 0; x < croppedWidthZ; x++ )
    for( y = 0; y < croppedHeightZ; y++ )
        dZ[ x ][ y ] = Z[ croppedWidthZ − x − 1 ][ y ]                    (D-19)
```

Where croppedWidthZ is the width of the component Z of the cropped decoded picture in samples, croppedHeightZ is the height of the component Z of the cropped decoded picture in samples, and Z[x][y] and dZ[x][y] are the sample value before and after the horizontal flipping, respectively, for the sample at the location (x, y) of the component Z of the cropped decoded picture.

ver_flip equal to 1 indicates that the cropped decoded picture should be flipped vertically (in addition to any horizontal flipping when hor_flip is equal to 1) for display. ver_flip equal to 0 indicates that the decoded picture should not be flipped vertically.

When ver_flip is equal to 1, the cropped decoded picture should be flipped as follows for each component Z equal to Y, Cb and Cr, letting dZ be the final cropped array of output samples for the component Z:

```
for( x = 0; x < croppedWidthZ; x++ )
    for( y = 0; y < croppedHeightZ; y++ )
        dZ[ x ][ y ] = Z[ x ][ croppedWidthZ − y − 1 ]                    (D-20)
```

Where croppedWidthZ is the width of the component Z of the cropped decoded picture in samples, croppedHeightZ is the height of the component Z of the cropped decoded picture in samples, and Z[x][y] and dZ[x][y] are the sample value before and after the vertical flipping, respectively, for the sample at the location (x, y) of the component Z of the cropped decoded picture.

anticlockwise_rotation specifies the recommended anticlockwise rotation of the decoded picture (after applying horizontal or vertical flipping when hor_flip or ver_flip is set) prior to display. The decoded picture should be rotated by 360 * anticlockwise_rotation ÷ 2^{16} degrees (2 * π * anticlockwise_rotation ÷ 2^{16} radians, where π is Archimedes' constant 3.141 592 653 589 793...) in the anticlockwise direction prior to display. For example, anticlockwise_rotation equal to 0

indicates no rotation and anticlockwise_rotation equal to 16 384 indicates 90 degrees (π ÷ 2 radians) rotation in the anticlockwise direction.

NOTE – It is possible for equivalent transformations to be expressed in multiple ways using these syntax elements. For example, the combination of having both hor_flip and ver_flip equal to 1 with anticlockwise_rotation equal to 0 can alternatively be expressed by having both hor_flip and ver_flip equal to 1 with anticlockwise_rotation equal to 0x8000, and the combination of hor_flip equal to 1 with ver_flip equal to 0 and anticlockwise_rotation equal to 0 can alternatively be expressed by having hor_flip equal to 0 with ver_flip equal to 1 and anticlockwise_rotation equal to 0x8000.

display_orientation_persistence_flag specifies the persistence of the display orientation SEI message for the current layer.

display_orientation_persistence_flag equal to 0 specifies that the display orientation SEI message applies to the current decoded picture only.

Let picA be the current picture. display_orientation_persistence_flag equal to 1 specifies that the display orientation SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.
– The bitstream ends.
– A picture picB in the current layer in an access unit containing a display orientation SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

D.3.18 Green metadata SEI message semantics

The semantics for this SEI message are specified in ISO/IEC 23001-11 (Green metadata). Green metadata facilitates reduced power consumption in decoders, encoders, displays and in media selection.

D.3.19 Structure of pictures information SEI message semantics

The structure of pictures information SEI message provides information for a list of entries, some of which correspond to the target picture set that consists of a series of pictures starting from the current picture until the last picture in decoding order in the current layer in the CLVS.

The first entry in the structure of pictures information SEI message corresponds to the current picture. When there is a picture in the target picture set that has PicOrderCntVal equal to the variable entryPicOrderCnt[i] as specified below, the entry i corresponds to a picture in the target picture set. The decoding order of the pictures in the target picture set that correspond to entries in the structure of pictures information SEI message corresponds to increasing values of i in the list of entries.

Any picture picB in the target picture set that has PicOrderCntVal equal to entryPicOrderCnt[i] for any i in the range of 0 to num_entries_in_sop_minus1, inclusive, where PicOrderCntVal is the value of PicOrderCntVal of picB immediately after the invocation of the decoding process for picture order count for picB, shall correspond to an entry in the list of entries.

The structure of pictures information SEI message shall not be present in a CVS and applicable for a layer for which the active SPS has long_term_ref_pics_present_flag equal to 1 or num_short_term_ref_pic_sets equal to 0.

The structure of pictures information SEI message shall not be present in any access unit that has TemporalId greater than 0 or contains a RASL, RADL or SLNR picture in the current layer. Any picture in the target picture set that corresponds to an entry other than the first entry described in the structure of pictures information SEI message shall not be an IRAP picture.

sop_seq_parameter_set_id indicates and shall be equal to the sps_seq_parameter_set_id value of the active SPS. The value of sop_seq_parameter_set_id shall be in the range of 0 to 15, inclusive.

num_entries_in_sop_minus1 plus 1 specifies the number of entries in the structure of pictures information SEI message. num_entries_in_sop_minus1 shall be in the range of 0 to 1 023, inclusive.

sop_vcl_nut[i], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the nal_unit_type value of the picture corresponding to the i-th entry.

sop_temporal_id[i], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the TemporalId value of the picture corresponding to the i-th entry. The value of 7 for sop_temporal_id[i] is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore structure of pictures information SEI messages that contain the value 7 for sop_temporal_id[i].

sop_short_term_rps_idx[i], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the index, into the list of candidate short-term RPSs included in the active SPS, of the candidate short-term RPS used by the picture corresponding to the i-th entry for derivation of the short-term reference picture set. sop_short_term_rps_idx[i] shall be in the range of 0 to num_short_term_ref_pic_sets − 1, inclusive.

sop_poc_delta[i] is used to specify the value of the variable entryPicOrderCnt[i] for the i-th entry described in the structure of pictures information SEI message. sop_poc_delta[i] shall be in the range of (−MaxPicOrderCntLsb) / 2 + 1 to MaxPicOrderCntLsb / 2 − 1, inclusive.

The variable entryPicOrderCnt[i] is derived as follows:

```
entryPicOrderCnt[ 0 ] = PicOrderCnt( currPic )
for( i = 1; i  <=  num_entries_in_sop_minus1; i++ )
    entryPicOrderCnt[ i ] = entryPicOrderCnt[ i − 1 ] + sop_poc_delta[ i ]                    (D-21)
```

where currPic is the current picture.

D.3.20 Decoded picture hash SEI message semantics

This message provides a hash for each colour component of the current decoded picture.

NOTE 1 – The decoded picture hash SEI message is a suffix SEI message and cannot be contained in a scalable nesting SEI message.

Prior to computing the hash, the decoded picture data are arranged into one or three strings of bytes called pictureData[cIdx] of lengths dataLen[cIdx] as follows:

```
for( cIdx = 0; cIdx < ( chroma_format_idc  = =  0 ) ? 1 : 3; cIdx++ ) {
    if( cIdx  = =  0 ) {
        compWidth[ cIdx ] = pic_width_in_luma_samples
        compHeight[ cIdx ] = pic_height_in_luma_samples
        compDepth[ cIdx ] = BitDepthY
    } else {
        compWidth[ cIdx ] = pic_width_in_luma_samples / SubWidthC
        compHeight[ cIdx ] = pic_height_in_luma_samples / SubHeightC
        compDepth[ cIdx ] = BitDepthC                                                    (D-22)
    }
    iLen = 0
    for( i = 0; i < compWidth[ cIdx ] * compHeight[ cIdx ]; i++ ) {
        pictureData[ cIdx ][ iLen++ ] = component[ cIdx ][ i ] & 0xFF
        if( compDepth[ cIdx ] > 8 )
            pictureData[ cIdx ][ iLen++ ] = component[ cIdx ][ i ]  >>  8
    }
    dataLen[ cIdx ] = iLen
}
```

where component[cIdx][i] is an array in raster scan of decoded sample values in two's complement representation.

hash_type indicates the method used to calculate the checksum according to Table D.10. Values of hash_type that are not listed in Table D.10 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore decoded picture hash SEI messages that contain reserved values of hash_type.

Table D.10 – Interpretation of hash_type

hash_type	Method
0	MD5 (IETF RFC 1321)
1	CRC
2	Checksum

picture_md5[cIdx][i] is the 16-byte MD5 hash of the cIdx-th colour component of the decoded picture. The value of picture_md5[cIdx][i] shall be equal to the value of digestVal[cIdx] obtained as follows, using the MD5 functions defined in IETF RFC 1321:

```
MD5Init( context )
MD5Update( context, pictureData[ cIdx ], dataLen[ cIdx ] )                    (D-23)
MD5Final( digestVal[ cIdx ], context )
```

picture_crc[cIdx] is the cyclic redundancy check (CRC) of the colour component cIdx of the decoded picture. The value of picture_crc[cIdx] shall be equal to the value of crcVal[cIdx] obtained as follows:

```
crc = 0xFFFF
pictureData[ cIdx ][  dataLen[ cIdx ] ] = 0
pictureData[ cIdx ][  dataLen[ cIdx ] + 1 ] = 0
for( bitIdx = 0; bitIdx < ( dataLen[ cIdx ]  + 2 ) * 8; bitIdx++ ) {
    dataByte = pictureData[ cIdx ][ bitIdx  >>  3 ]
    crcMsb = ( crc  >>  15 ) & 1
    bitVal = ( dataByte  >>  ( 7 − ( bitIdx & 7 ) ) ) & 1
    crc = ( ( ( crc  <<  1 ) + bitVal ) & 0xFFFF ) ^ ( crcMsb * 0x1021 )
}
crcVal[ cIdx ] = crc                                                           (D-24)
```

NOTE 2 – The same CRC specification is found in Rec. ITU-T H.271.

picture_checksum[cIdx] is the checksum of the colour component cIdx of the decoded picture. The value of picture_checksum[cIdx] shall be equal to the value of checksumVal[cIdx] obtained as follows:

```
sum = 0
for( y = 0; y < compHeight[ cIdx ]; y++ )
    for( x = 0; x < compWidth[ cIdx ]; x++ ) {
        xorMask = ( x & 0xFF ) ^ ( y & 0xFF ) ^ ( x  >>  8 ) ^ ( y  >>  8 )
        sum = ( sum + ( ( component[ cIdx ][ y * compWidth[ cIdx ] + x ] & 0xFF ) ^ xorMask ) )
&
                0xFFFFFFFF
        if( compDepth[ cIdx ] > 8 )
            sum = ( sum + ( ( component[ cIdx ][ y * compWidth[ cIdx ] + x ]  >>  8 ) ^ xorMask )
) &
                    0xFFFFFFFF
    }
checksumVal[ cIdx ] = sum                                                      (D-25)
```

D.3.21 Active parameter sets SEI message semantics

The active parameter sets SEI message indicates which VPS is active for the VCL NAL units of the access unit associated with the SEI message. The SEI message may also provide information on which SPS is active for the VCL NAL units of the access unit associated with the SEI message and other information related to parameter sets.

active_video_parameter_set_id indicates and shall be equal to the value of the vps_video_parameter_set_id of the VPS that is referred to by the VCL NAL units of the access unit associated with the SEI message. The value of active_video_parameter_set_id shall be in the range of 0 to 15, inclusive.

self_contained_cvs_flag equal to 1 indicates that each parameter set that is (directly or indirectly) referenced by any VCL NAL unit of the CVS that is not a VCL NAL unit of a RASL picture is present within the CVS at a position that precedes, in decoding order, any NAL unit that (directly or indirectly) references the parameter set. self_contained_cvs_flag equal to 0 indicates that this property may or may not apply.

no_parameter_set_update_flag equal to 1 indicates that there is no parameter set update in the CVS, i.e., each VPS in the CVS is an exact copy of the previous VPS in decoding order in the bitstream that has the same value of vps_video_parameter_set_id, when present; each SPS in the CVS is an exact copy of the previous SPS in decoding order in the bitstream that has the same value of sps_seq_parameter_set_id, when present; and each PPS in the CVS is an exact copy of the previous PPS in decoding order in the bitstream that has the same value of pps_pic_parameter_set_id, when present. no_parameter_set_update_flag equal to 0 indicates that there may or may not be parameter set update in the CVS.

NOTE 1 – If no_parameter_set_update_flag equal to 1 is indicated for each CVS in a bitstream, i.e., there is no parameter set update in the bitstream, it is possible to transmit all parameter sets out-of-band before sending the first VCL NAL unit of the bitstream or to place all parameter sets at the beginning of the bitstream. Otherwise, out-of-band transmission of all parameter sets before sending VCL NAL units is not possible.

num_sps_ids_minus1 plus 1 shall be less than or equal to the number of SPSs that are referred to by the VCL NAL units of the access unit associated with the active parameter sets SEI message. The value of num_sps_ids_minus1 shall be in the range of 0 to 15, inclusive.

active_seq_parameter_set_id[i] indicates a value of the sps_seq_parameter_set_id of the SPS that may be referred to by any VCL NAL unit of the access unit associated with the SEI message. When vps_base_layer_internal_flag is equal to 1 and vps_base_layer_available_flag is equal to 1, active_seq_parmeter_set_id[0] indicates and shall be equal to the value of the sps_seq_parameter_set_id of the SPS that is referred to by the VCL NAL units with nuh_layer_id equal to 0 in the access unit associated with the SEI message. The value of active_seq_parameter_set_id[i] shall be in the range of 0 to 15, inclusive.

layer_sps_idx[i] indicates that the sps_seq_parameter_set_id value of the SPS that is referred to by the VCL NAL units with nuh_layer_id equal to layer_id_in_nuh[i] in the access unit associated with the SEI message, if any, is equal to active_seq_parameter_set_id[layer_sps_idx[i]].

NOTE 2 – When decoding a single-layer bitstream or only the base layer of a multi-layer bitstream, the decoder does not need to parse layer_sps_idx[i] syntax elements, because they do not affect the decoding of the base layer.

D.3.22 Decoding unit information SEI message semantics

The decoding unit information SEI message provides CPB removal delay information for the decoding unit associated with the SEI message.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, the following applies for the decoding unit information SEI message syntax and semantics:

– The syntax elements sub_pic_hrd_params_present_flag, sub_pic_cpb_params_in_pic_timing_sei_flag and dpb_output_delay_du_length_minus1, and the variable CpbDpbDelaysPresentFlag are found in or derived from syntax elements in the hrd_parameters( ) syntax structure that is applicable to at least one of the operation points to which the decoding unit information SEI message applies.

– The bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with any of the operation points to which the decoding unit information SEI message applies.

The presence of decoding unit information SEI messages for an operation point including the base layer is specified as follows:

– If CpbDpbDelaysPresentFlag is equal to 1, sub_pic_hrd_params_present_flag is equal to 1 and sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 0, one or more decoding unit information SEI messages applicable to the operation point shall be associated with each decoding unit in the CVS.
– Otherwise, if CpbDpbDelaysPresentFlag is equal to 1, sub_pic_hrd_params_present_flag is equal to 1 and sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 1, one or more decoding unit information SEI messages applicable to the operation point may or may not be associated with each decoding unit in the CVS.
– Otherwise (CpbDpbDelaysPresentFlag is equal to 0 or sub_pic_hrd_params_present_flag is equal to 0), in the CVS there shall be no decoding unit that is associated with a decoding unit information SEI message applicable to the operation point.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, the set of NAL units associated with a decoding unit information SEI message consists, in decoding order, of the SEI NAL unit containing the decoding unit information SEI message and all subsequent NAL units in the access unit up to but not including any subsequent SEI NAL unit containing a decoding unit information SEI message with a different value of decoding_unit_idx. Each decoding unit shall include at least one VCL NAL unit. All non-VCL NAL units associated with a VCL NAL unit shall be included in the decoding unit containing the VCL NAL unit.

decoding_unit_idx specifies the index, starting from 0, to the list of decoding units in the current access unit, of the decoding unit associated with the decoding unit information SEI message. The value of decoding_unit_idx shall be in the range of 0 to PicSizeInCtbsY − 1, inclusive.

A decoding unit identified by a particular value of duIdx includes and only includes all NAL units associated with all decoding unit information SEI messages that have decoding_unit_idx equal to duIdx. Such a decoding unit is also referred to as associated with the decoding unit information SEI messages having decoding_unit_idx equal to duIdx.

For any two decoding units duA and duB in one access unit with decoding_unit_idx equal to duIdxA and duIdxB, respectively, where duIdxA is less than duIdxB, duA shall precede duB in decoding order.

A NAL unit of one decoding unit shall not be present, in decoding order, between any two NAL units of another decoding unit.

du_spt_cpb_removal_delay_increment specifies the duration, in units of clock sub-ticks, between the nominal CPB times of the last decoding unit in decoding order in the current access unit and the decoding unit associated with the decoding unit information SEI message. This value is also used to calculate an earliest possible time of arrival of decoding unit data into the CPB for the HSS, as specified in Annex C or clause F.13. The syntax element is represented by a fixed length code whose length in bits is given by du_cpb_removal_delay_increment_length_minus1 + 1. When the decoding unit associated with the decoding unit information SEI message is the last decoding unit in the current access unit, the value of du_spt_cpb_removal_delay_increment shall be equal to 0.

dpb_output_du_delay_present_flag equal to 1 specifies the presence of the pic_spt_dpb_output_du_delay syntax element in the decoding unit information SEI message. dpb_output_du_delay_present_flag equal to 0 specifies the absence of the pic_spt_dpb_output_du_delay syntax element in the decoding unit information SEI message.

pic_spt_dpb_output_du_delay is used to compute the DPB output time of the picture when SubPicHrdFlag is equal to 1. It specifies how many sub clock ticks to wait after removal of the last decoding unit in an access unit from the CPB before the decoded picture is output from the DPB. When not present, the value of pic_spt_dpb_output_du_delay is inferred to be equal to pic_dpb_output_du_delay.

The length of the syntax element pic_spt_dpb_output_du_delay is given in bits by dpb_output_delay_du_length_minus1 + 1.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, it is a requirement of bitstream conformance that all decoding unit information SEI messages that are associated with the same access unit, apply to the same operation point, and have dpb_output_du_delay_present_flag equal to 1 shall have the same value of pic_spt_dpb_output_du_delay.

The output time derived from the pic_spt_dpb_output_du_delay of any picture that is output from an output timing conforming decoder shall precede the output time derived from the pic_spt_dpb_output_du_delay of all pictures in any subsequent CVS in decoding order.

The picture output order established by the values of this syntax element shall be the same order as established by the values of PicOrderCntVal.

For pictures that are not output by the "bumping" process because they precede, in decoding order, an IRAP picture with NoRaslOutputFlag equal to 1 that has no_output_of_prior_pics_flag equal to 1 or inferred to be equal to 1, the output times derived from pic_spt_dpb_output_du_delay shall be increasing with increasing value of PicOrderCntVal relative to all pictures within the same CVS.

For any two pictures in the CVS, the difference between the output times of the two pictures when SubPicHrdFlag is equal to 1 shall be identical to the same difference when SubPicHrdFlag is equal to 0.

D.3.23 Temporal sub-layer zero index SEI message semantics

The temporal sub-layer zero index SEI message provides information that can assist the decoder for detection of missing coded pictures that have TemporalId equal to 0 and are not RASL, RADL or SLNR pictures.

When a temporal sub-layer zero index SEI message is present in the current access unit and applies to the current layer and the current picture is not an IRAP picture, a temporal sub-layer zero index SEI message that applies to the current layer shall also be present in the preceding access unit in decoding order with TemporalId equal to 0 and containing a picture that is not a RASL, RADL or SLNR picture in the current layer.

NOTE – Encoders should be cautious when setting discardable_flag equal to 1 for any picture with TemporalId equal to 0, as based on temporal sub-layer zero index SEI messages decoders may consider intentionally discarded pictures with TemporalId equal to 0 and discardable_flag equal to 1 as lost and take unnecessary error recovery actions, such as requesting retransmission of a missing picture with TemporalId equal to 0.

Let ReferencedTsl0Pic denote a picture that has TemporalId equal to 0 and is not a RASL, RADL or SLNR picture.

temporal_sub_layer_zero_idx indicates a temporal sub-layer zero index as follows:

– If the current picture is a ReferencedTsl0Pic, temporal_sub_layer_zero_idx indicates the temporal sub-layer zero index for the current picture.

– Otherwise, temporal_sub_layer_zero_idx indicates the temporal sub-layer zero index of the preceding picture in the current layer in decoding order that is a ReferencedTsl0Pic.

When the bitstream contains a preceding access unit in decoding order that contained a ReferencedTsl0Pic, and that preceding access unit has an associated temporal sub-layer zero index SEI message seiMsgB that applies to the current layer, the variable prevTsl0Idx is set equal to the value of temporal_sub_layer_zero_idx of seiMsgB.

The following constraints apply to the value of temporal_sub_layer_zero_idx:

– If the current picture is an IRAP picture, temporal_sub_layer_zero_idx shall be equal to 0.

– Otherwise, the following applies:

 – If the current picture is a ReferencedTsl0Pic, temporal_sub_layer_zero_idx shall be equal to (prevTsl0Idx + 1) % 256.

 – Otherwise, temporal_sub_layer_zero_idx shall be equal to prevTsl0Idx.

irap_pic_id is an IRAP picture identifier for the current layer. When the current picture is not the first picture in the current layer in the bitstream in decoding order and the preceding IRAP picture in the current layer in decoding order has an associated temporal sub-layer zero index SEI message, the following constraints apply to the value of irap_pic_id:

– If the current picture is an IRAP picture, irap_pic_id shall differ in value from the value of irap_pic_id of the temporal sub-layer zero index SEI message of the preceding IRAP picture in the current layer in decoding order.

 NOTE – It is suggested for the value of irap_pic_id to be set to a random value (subject to the constraints specified herein), to minimize the likelihood of duplicate values appearing in a layer in the bitstream due to picture losses or splicing operations.

– Otherwise, irap_pic_id shall be equal to the value of irap_pic_id of the temporal sub-layer zero index SEI message associated with the preceding IRAP picture in the current layer in decoding order.

D.3.24 Scalable nesting SEI message semantics

The scalable nesting SEI message provides a mechanism to associate SEI messages with bitstream subsets corresponding to various operation points or with specific layers or sub-layers.

A scalable nesting SEI message contains one or more SEI messages.

NOTE – A scalable nesting SEI message can only be the last SEI message in the SEI NAL unit containing the scalable nesting SEI message. The sei_payload( ) of the scalable nesting SEI message cannot contain any reserved_payload_extension_data.

It is a requirement of bitstream conformance that the following restrictions apply on containing of SEI messages in a scalable nesting SEI message:

– An SEI message that has payloadType equal to 129 (active parameter sets), 132 (decoded picture hash) or 133 (scalable nesting) shall not be contained in a scalable nesting SEI message.

– When a scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, the scalable nesting SEI message shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information).

bitstream_subset_flag equal to 0 specifies that the SEI messages contained in the scalable nesting SEI message apply to specific layers or sub-layers. bitstream_subset_flag equal to 1 specifies that the SEI messages contained in the scalable nesting SEI message apply to one or more sub-bitstreams as specified below.

Depending on the value of bitstream_subset_flag, the layers or sub-layers, or the operation points to which the SEI messages contained in the scalable nesting SEI message apply are specified by deriving the lists nestingLayerIdList[i] and the variables MaxTemporalId[i] based on syntax element values as specified below.

It is a requirement of bitstream conformance that the following restrictions apply on the value of bitstream_subset_flag:

– When the scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, bitstream_subset_flag shall be equal to 1.

– When the scalable nesting SEI message contains an SEI message that has payloadType equal to any value among SingleLayerSeiList, bitstream_subset_flag shall be equal to 0.

nesting_op_flag equal to 0 specifies that the list nestingLayerIdList[0] is specified by all_layers_flag and, when present, nesting_layer_id[i] for all i values in the range of 0 to nesting_num_layers_minus1, inclusive, and that the variable MaxTemporalId[0] is specified by nesting_no_op_max_temporal_id_plus1. nesting_op_flag equal to 1 specifies that the list nestingLayerIdList[i] and the variable MaxTemporalId[i] are specified by nesting_num_ops_minus1, default_op_flag, nesting_max_temporal_id_plus1[i], when present, and nesting_op_idx[i], when present.

default_op_flag equal to 1 specifies that MaxTemporalId[0] is equal to nuh_temporal_id_plus1 of the current SEI NAL unit minus 1 and that nestingLayerIdList[0] contains all integer values in the range of 0 to nuh_layer_id of the current SEI NAL unit, inclusive, in increasing order of the values.

nesting_num_ops_minus1 plus 1 minus default_op_flag specifies the number of the following nesting_op_idx[i] syntax elements. The value of nesting_num_ops_minus1 shall be in the range of 0 to 1 023, inclusive.

If nesting_op_flag is equal to 0, the variable nestingNumOps is set equal to 1. Otherwise, the variable nestingNumOps is set equal to nesting_num_ops_minus1 + 1.

nesting_max_temporal_id_plus1[i] is used to specify the variable MaxTemporalId[i]. The value of nesting_max_temporal_id_plus1[i] shall be greater than or equal to nuh_temporal_id_plus1 of the current SEI NAL unit. The variable MaxTemporalId[i] is set equal to nesting_max_temporal_id_plus1[i] − 1.

nesting_op_idx[i] is used to specify the list nestingLayerIdList[i]. The value of nesting_op_idx[i] shall be in the range of 0 to 1 023, inclusive.

The list nestingLayerIdList[i] is set equal to the OpLayerIdList of the nesting_op_idx[i]-th layer set specified by the active VPS.

It is a requirement of bitstream conformance that when nesting_op_flag is equal to 1 and nesting_op_idx[i] is greater than vps_num_layer_sets_minus1 for any value of i in the range of default_op_flag to nesting_num_ops_minus1, inclusive, the value of nuh_layer_id of the SEI NAL unit containing the scalable nesting SEI message shall be greater than 0.

all_layers_flag equal to 0 specifies that the list nestingLayerIdList[0] is specified by nesting_layer_id[i] for all i values in the range of 0 to nesting_num_layers_minus1, inclusive. all_layers_flag equal to 1 specifies that the list nestingLayerIdList[0] consists of all values of nuh_layer_id present in the current access unit that are greater than or equal to nuh_layer_id of the current SEI NAL unit, in increasing order of the values.

When nesting_op_flag is equal to 0 and all_layers_flag is equal to 1, MaxTemporalId[0] is set equal to 6.

nesting_no_op_max_temporal_id_plus1 minus 1 specifies the value of MaxTemporalId[0] when nesting_op_flag is equal to 0 and all_layers_flag is equal to 0. The value of nesting_no_op_max_temporal_id_plus1 shall not be equal to 0.

nesting_num_layers_minus1 plus 1 specifies the number of the following nesting_layer_id[i] syntax elements. The value of nesting_num_layers_minus1 shall be in the range of 0 to 63, inclusive.

nesting_layer_id[i] specifies the i-th nuh_layer_id value included in the list nestingLayerIdList[0].

For any i and j in the range of 0 to nesting_num_layers_minus1, inclusive, with i less than j, nesting_layer_id[i] shall be less than nesting_layer_id[j].

The list nestingLayerIdList[0] is set to consist of nesting_layer_id[i] for all i values in the range of 0 to nesting_num_layers_minus1, inclusive, in increasing order of i values.

When bitstream_subset_flag is equal to 0, the following applies:

– The SEI messages contained in the scalable nesting SEI message apply to the sets of layers or sub-layers subLayerSet[i] for all i values in the range of 0 to nestingNumOps − 1, inclusive, where the VCL NAL units of the layers or sub-layers in each set subLayerSet[i] have nuh_layer_id values that are included in the list nestingLayerIdList[i] and TemporalId values that are in the range of the TemporalId of the current SEI NAL unit to MaxTemporalId[i], inclusive.

– When a scalable-nested SEI message has payloadType equal to any value among VclAssociatedSeiList, the value of TemporalId of the SEI NAL unit containing the scalable nesting SEI message shall be equal to 0 and MaxTemporalId[i] for all values of i in the range of 0 to nestingNumOps − 1, inclusive, shall be equal to 6.

– When a scalable-nested SEI message has payloadType equal to any value among VclAssociatedSeiList and the value of nestingNumOps is greater than 0, the scalable-nested SEI message applies to all layers for which each nuh_layer_id is included in at least one of the lists nestingLayerIdList[i] with i ranging from 0 to nestingNumOps − 1, inclusive.

When bitstream_subset_flag is equal to 1, the SEI messages contained in the scalable nesting SEI message apply to the operation points corresponding to the sub-bitstreams subBitstream[i] for all i values in the range of 0 to nestingNumOps − 1, inclusive, where each sub-bitstream subBitstream[i] is derived as follows:

– If nestingLayerIdList[i][0] is equal to 0 and vps_base_layer_internal_flag is equal to 1, the sub-bitstream subBitstream[i] is the output of the sub-bitstream extraction process of clause 10 with the bitstream, MaxTemporalId[i] and nestingLayerIdList[i] as inputs.

– Otherwise, if nestingLayerIdList[i][0] is equal to 0 and vps_base_layer_internal_flag is equal to 0, the sub-bitstream subBitstream[i] is the output of the sub-bitstream extraction process of clause F.10.1 with the bitstream, MaxTemporalId[i] and nestingLayerIdList[i] as inputs.

– Otherwise, the sub-bitstream subBitstream[i] is the output of the sub-bitstream extraction process of clause F.10.3 with the bitstream, MaxTemporalId[i] and nestingLayerIdList[i] as inputs.

nesting_zero_bit shall be equal to 0.

D.3.25 Region refresh information SEI message semantics

The region refresh information SEI message indicates whether the slice segments that the current SEI message applies to belong to a refreshed region of the current picture (as defined below).

An access unit that is not an IRAP access unit and that contains a recovery point SEI message is referred to as a gradual decoding refresh (GDR) access unit, and its corresponding picture is referred to as a GDR picture. The access unit corresponding to the indicated recovery point picture is referred to as the recovery point access unit.

If there is a picture that follows the GDR picture in decoding order in the CVS and that has PicOrderCntVal equal to the PicOrderCntVal of the GDR picture plus the value of recovery_poc_cnt in the recovery point SEI message, let the variable lastPicInSet be the recovery point picture. Otherwise, let lastPicInSet be the picture that immediately precedes the recovery point picture in output order. The picture lastPicInSet shall not precede the GDR picture in decoding order.

Let gdrPicSet be the set of pictures starting from a GDR picture to the picture lastPicInSet, inclusive, in output order. When the decoding process is started from a GDR access unit, the refreshed region in each picture of the gdrPicSet is indicated to be the region of the picture that is correct or approximately correct in content, and, when lastPicInSet is the recovery point picture, the refreshed region in lastPicInSet covers the entire picture.

The slice segments to which a region refresh information SEI message applies consist of all slice segments within the access unit that follow the SEI NAL unit containing the region refresh information SEI message and precede the next SEI NAL unit containing a region refresh information SEI message (if any) in decoding order. These slice segments are referred to as the slice segments associated with the region refresh information SEI message.

Let gdrAuSet be the set of access units corresponding to gdrPicSet. A gdrAuSet and the corresponding gdrPicSet are referred to as being associated with the recovery point SEI message contained in the GDR access unit.

Region refresh information SEI messages shall not be present in an access unit unless the access unit is included in a gdrAuSet associated with a recovery point SEI message. When any access unit that is included in a gdrAuSet contains one or more region refresh information SEI messages, all access units in the gdrAuSet shall contain one or more region refresh information SEI messages.

refreshed_region_flag equal to 1 indicates that the slice segments associated with the current SEI message belong to the refreshed region in the current picture. refreshed_region_flag equal to 0 indicates that the slice segments associated with the current SEI message may not belong to the refreshed region in the current picture.

When one or more region refresh information SEI messages are present in an access unit and the first slice segment of the access unit in decoding order does not have an associated region refresh information SEI message, the value of refreshed_region_flag for the slice segments that precede the first region refresh information SEI message is inferred to be equal to 0.

When lastPicInSet is the recovery point picture, and any region refresh SEI message is included in a recovery point access unit, the first slice segment of the access unit in decoding order shall have an associated region refresh SEI message, and the value of refreshed_region_flag shall be equal to 1 in all region refresh SEI messages in the access unit.

When one or more region refresh information SEI messages are present in an access unit, the refreshed region in the picture is specified as the set of CTUs in all slice segments of the access unit that are associated with region refresh information SEI messages that have refreshed_region_flag equal to 1. Other slice segments belong to the non-refreshed region of the picture.

It is a requirement of bitstream conformance that when a dependent slice segment belongs to the refreshed region, the preceding slice segment in decoding order shall also belong to the refreshed region.

Let gdrRefreshedSliceSegmentSet be the set of all slice segments that belong to the refreshed regions in the gdrPicSet. When a gdrAuSet contains one or more region refresh information SEI messages, it is a requirement of bitstream conformance that the following constraints all apply:

– The refreshed region in the first picture included in the corresponding gdrPicSet in decoding order that contains any refreshed region shall contain only coding units that are coded in an intra coding mode.
– For each picture included in the gdrPicSet, the syntax elements in gdrRefreshedSliceSegmentSet shall be constrained such that no samples or motion vector values outside of gdrRefreshedSliceSegmentSet are used for inter prediction in the decoding process of any samples within gdrRefreshedSliceSegmentSet.
– For any picture that follows the picture lastPicInSet in output order, the syntax elements in the slice segments of the picture shall be constrained such that no samples or motion vector values outside of gdrRefreshedSliceSegmentSet are used for inter prediction in the decoding process of the picture other than those of the other pictures that follow the picture lastPicInSet in output order.

reserved_sei_message_payload_byte is a byte reserved for future use by ITU-T | ISO/IEC.

D.3.26 No display SEI message semantics

The no display SEI message indicates that the current picture should not be displayed.

D.3.27 Time code SEI message semantics

The time code SEI message provides time code information similar to that defined by SMPTE ST 12-1 (2014) for field(s) or frame(s) of the current picture. When vps_timing_info_present_flag is equal to 1, it also defines a time offset for use in calculating a reference clock timestamp from the time code syntax elements to indicate a time of origin, capture, or alternative ideal display.

num_clock_ts specifies the number of sets of clock timestamp syntax elements that may be present for the current picture, the presence of each of which is specified by a syntax flag clock_timestamp_flag[i] for a corresponding value of i. When frame_field_info_present_flag is equal to 1, the i-th set of clock timestamp syntax elements applies to the i-th field or frame associated with the indicated display of the current picture in the order specified by pic_struct in Table D.2. The value of num_clock_ts shall not be equal to 0.

When field_seq_flag is equal to 1, the value of num_clock_ts shall be equal to 1.

When frame_field_info_present_flag is equal to 1, the value of num_clock_ts shall be equal to the value of DeltaToDivisor specified in Table E.7.

When vps_timing_info_present_flag is equal to 1, the contents of the clock timestamp syntax elements indicate a time of origin, capture, or alternative ideal display. This indicated time is computed as

$$\begin{aligned}\text{clockTimestamp}[\,i\,] = (\,(\,\text{hH} * 60 + \text{mM}\,) * 60 + \text{sS}\,) * \text{vui\_time\_scale} + \\ \text{nFrames} * (\,\text{vui\_num\_units\_in\_tick} * (\,1 + \text{units\_field\_based\_flag}[\,i\,]\,)\,) + \text{tOffset}\end{aligned} \tag{D-26}$$

in units of clock ticks of a clock with clock frequency equal to vui_time_scale Hz, relative to some unspecified point in time for which clockTimestamp[i] would be equal to 0. Output order and DPB output timing are not affected by the value of clockTimestamp[i]. When frame_field_info_present_flag is equal to 1 and two or more pictures with pic_struct equal to 0 are consecutive in output order and have equal values of clockTimestamp[i], the indication is that the pictures represent the same content and that the last such picture in output order is the preferred representation.

NOTE 1 – clockTimestamp[i] time indications may aid display on devices with refresh rates other than those well-matched to DPB output times. However, the time code SEI message does not affect the output order and DPB output timing defined in this Specification.

clock_timestamp_flag[i] equal to 1 indicates that associated set of clock timestamp syntax elements are present for the i-th set of clock timestamp syntax elements of the current picture. clock_timestamp_flag[i] equal to 0 indicates that the associated set of clock timestamp syntax elements is not present. When vps_timing_info_present_flag is equal to 1, num_clock_ts is greater than 1 and clock_timestamp_flag[i] is equal to 1 for more than one value of i, the value of clockTimestamp[i] shall be non-decreasing with increasing value of i.

For the i-th set of clock timestamp syntax elements of the current picture, the previous set of clock timestamp syntax elements in decoding order (or in output order, respectively), is defined as follows:

– If i is equal to 0, the previous set of clock timestamp syntax elements in decoding order (or in output order, respectively) is the set of syntax elements for the highest value of i for the previous picture in decoding order (or in output order, respectively) that contained a time code SEI message.

– Otherwise, the previous set of clock timestamp syntax elements in decoding order (or in output order, respectively) is the (i − 1)-th set of clock timestamp syntax elements of the current picture.

units_field_based_flag[i] is used in calculating clockTimestamp[i], as specified in Equation D-26.

NOTE 2 – units_field_based_flag[i] should be the same for all values of i for all pictures in the CVS. When field_seq_flag is equal to 1 or frame_field_info_present_flag is equal to 1 and pic_struct is in the range of 1 to 6 or 9 to 12, inclusive, units_field_based_flag[i] should be equal to 1.

counting_type[i] specifies the method of dropping values of the n_frames[i] syntax element as specified in Table D.11.

NOTE 3 – counting_type[i] should be the same for all values of i for all pictures in the CVS.

Table D.11 – Definition of counting_type[i] values

Value	Interpretation

0	No dropping of n_frames[i] count values and no use of time_offset_value[i]
1	No dropping of n_frames[i] count values
2	Dropping of individual zero values of n_frames[i] count
3	Dropping of individual values of n_frames[i] count equal to MaxFPS − 1
4	Dropping of the two lowest (value 0 and 1) n_frames[i] counts when seconds_value[i] is equal to 0 and minutes_value[i] is not an integer multiple of 10
5	Dropping of unspecified individual n_frames[i] count values
6	Dropping of unspecified numbers of unspecified n_frames[i] count values
7..31	Reserved

full_timestamp_flag[i] equal to 1 specifies that the n_frames[i] syntax element is followed by seconds_value[i], minutes_value[i] and hours_value[i]. full_timestamp_flag[i] equal to 0 specifies that the n_frames[i] syntax element is followed by seconds_flag[i].

discontinuity_flag[i] equal to 0 indicates that the difference between the current value of clockTimestamp[i] and the value of clockTimestamp[i] computed from the previous set of clock timestamp syntax elements in output order can be interpreted as the time difference between the times of origin or capture of the associated frames or fields. discontinuity_flag[i] equal to 1 indicates that the difference between the current value of clockTimestamp[i] and the value of clockTimestamp[i] computed from the previous set of clock timestamp syntax elements in output order should not be interpreted as the time difference between the times of origin or capture of the associated frames or fields. When vps_timing_info_present_flag is equal to 1 and discontinuity_flag[i] is equal to 0, the value of clockTimestamp[i] shall be greater than or equal to the value of clockTimestamp[i] for the previous set of clock timestamp syntax elements in output order (when present).

cnt_dropped_flag[i] specifies the skipping of one or more values of n_frames[i] using the counting method specified by counting_type[i].

n_frames[i] specifies the value of nFrames used to compute clockTimestamp[i]. When vps_timing_info_present_flag is equal to 1, n_frames[i] shall be less than MaxFPS, where MaxFPS is specified by

$$\text{MaxFPS} = \text{Ceil}(\text{ vui\_time\_scale} \div ((1 + \text{units\_field\_based\_flag}[i]) * \text{vui\_num\_units\_in\_tick})) \quad \text{(D-27)}$$

NOTE 4 – n_frames[i] is a frame-based counter. To provide field-specific timing indications when the current picture is a frame, time_offset_value[i] should be used to indicate a distinct clockTimestamp[i] for each field of the frame.

When counting_type[i] is equal to 2 and cnt_dropped_flag[i] is equal to 1, n_frames[i] shall be equal to 1 and the value of n_frames[i] for the previous set of clock timestamp syntax elements in output order (when present) shall not be equal to 0 unless discontinuity_flag[i] is equal to 1.

NOTE 5 – When counting_type[i] is equal to 2, the need for increasingly large magnitudes of tOffset in Equation D-26 when using fixed non-integer frame rates (e.g., 12.5 frames per second with vui_time_scale equal to 50 and vui_num_units_in_tick equal to 2 and units_field_based_flag[i] equal to 0) can be avoided by occasionally skipping over the value n_frames[i] equal to 0 when counting (e.g., counting n_frames[i] from 0 to 12, then incrementing seconds_value[i] and counting n_frames[i] from 1 to 12, then incrementing seconds_value[i] and counting n_frames[i] from 0 to 12, etc.).

When vps_timing_info_present_flag is equal to 1, counting_type[i] is equal to 3 and cnt_dropped_flag[i] is equal to 1, n_frames[i] shall be equal to 0 and the value of n_frames[i] for the previous set of clock timestamp syntax elements in output order (when present) shall not be equal to MaxFPS − 1 unless discontinuity_flag[i] is equal to 1.

NOTE 6 – When counting_type[i] is equal to 3, the need for increasingly large magnitudes of tOffset in Equation D-26 when using fixed non-integer frame rates (e.g., 12.5 frames per second with vui_time_scale equal to 50 and vui_num_units_in_tick equal to 2 and units_field_based_flag[i] equal to 0) can be avoided by occasionally skipping over the value n_frames[i] equal to MaxFPS − 1 when counting (e.g., counting n_frames[i] from 0 to 12, then incrementing seconds_value[i] and counting n_frames[i] from 0 to 11, then incrementing seconds_value[i] and counting n_frames[i] from 0 to 12, etc.).

When counting_type[i] is equal to 4 and cnt_dropped_flag[i] is equal to 1, n_frames[i] shall be equal to 2 and seconds_value[i] shall be equal to zero and minutes_value[i] shall not be an integer multiple of 10 and n_frames[i] for the previous set of clock timestamp syntax elements in output order (when present) shall not be equal to 0 or 1 unless discontinuity_flag[i] is equal to 1.

NOTE 7 – When counting_type[i] is equal to 4, the need for increasingly large magnitudes of tOffset in Equation D-26 when using fixed non-integer frame rates (e.g., 30000÷1001 frames per second with vui_time_scale equal to 60 000 and vui_num_units_in_tick equal to 1 001 and units_field_based_flag[i] equal to 1) can be reduced by occasionally skipping over the values of n_frames[i]

equal to 0 and 1 when counting (e.g., counting n_frames[i] from 0 to 29, then incrementing seconds_value[i] and counting n_frames[i] from 0 to 29, etc., until the seconds_value[i] is zero and minutes_value[i] is not an integer multiple of ten, then counting n_frames[i] from 2 to 29, then incrementing seconds_value[i] and counting n_frames[i] from 0 to 29, etc.). This counting method is well known in the industry and is often referred to as "NTSC drop-frame" counting.

When vps_timing_info_present_flag is equal to 1, counting_type[i] is equal to 5 or 6 and cnt_dropped_flag[i] is equal to 1, n_frames[i] shall not be equal to 1 plus the value of n_frames[i] for the previous set of clock timestamp syntax elements in output order (when present) modulo MaxFPS unless discontinuity_flag[i] is equal to 1.

NOTE 8 – When counting_type[i] is equal to 5 or 6, the need for increasingly large magnitudes of tOffset in Equation D-26 when using fixed non-integer frame rates can be avoided by occasionally skipping over some values of n_frames[i] when counting. The specific values of n_frames[i] that are skipped are not specified when counting_type[i] is equal to 5 or 6.

seconds_flag[i] equal to 1 specifies that seconds_value[i] and minutes_flag[i] are present when full_timestamp_flag[i] is equal to 0. seconds_flag[i] equal to 0 specifies that seconds_value[i] and minutes_flag[i] are not present.

seconds_value[i] specifies the value of sS used to compute clockTimestamp[i]. The value of seconds_value[i] shall be in the range of 0 to 59, inclusive. When seconds_value[i] is not present, its value is inferred to be equal to the value of seconds_value[i] for the previous set of clock timestamp syntax elements in decoding order, and it is required that such a previous seconds_value[i] shall have been present.

minutes_flag[i] equal to 1 specifies that minutes_value[i] and hours_flag[i] are present when full_timestamp_flag[i] is equal to 0 and seconds_flag[i] is equal to 1. minutes_flag[i] equal to 0 specifies that minutes_value[i] and hours_flag[i] are not present.

minutes_value[i] specifies the value of mM used to compute clockTimestamp[i]. The value of minutes_value[i] shall be in the range of 0 to 59, inclusive. When minutes_value[i] is not present, its value is inferred to be equal to the value of minutes_value[i] for the previous set of clock timestamp syntax elements in decoding order, and it is required that such a previous minutes_value[i] shall have been present.

hours_flag[i] equal to 1 specifies that hours_value[i] is present when full_timestamp_flag[i] is equal to 0 and seconds_flag[i] is equal to 1 and minutes_flag[i] is equal to 1.

hours_value[i] specifies the value of hH used to compute clockTimestamp[i]. The value of hours_value[i] shall be in the range of 0 to 23, inclusive. When hours_value[i] is not present, its value is inferred to be equal to the value of hours_value[i] for the previous set of clock timestamp syntax elements in decoding order, and it is required that such a previous hours_value[i] shall have been present.

time_offset_length[i] greater than 0 specifies the length in bits of the time_offset_value[i] syntax element. time_offset_length[i] equal to 0 specifies that the time_offset_value[i] syntax element is not present. When counting_type[i] is equal to 0, time_offset_length[i] shall be equal to 0.

NOTE 9 – time_offset_length[i] should be the same for all values of i for all pictures in the CVS.

time_offset_value[i] specifies the value of tOffset used to compute clockTimestamp[i]. The number of bits used to represent time_offset_value[i] is equal to time_offset_length[i]. When time_offset_value[i] is not present, its value is inferred to be equal to 0.

D.3.28 Mastering display colour volume SEI message semantics

This SEI message identifies the colour volume (the colour primaries, white point, and luminance range) of a display considered to be the mastering display for the associated video content – e.g., the colour volume of a display that was used for viewing while authoring the video content. The described mastering display is a three-colour additive display system that has been configured to use the indicated mastering colour volume.

This SEI message does not identify the measurement methodologies and procedures used for determining the indicated values or provide any description of the mastering environment. It also does not provide information on colour transformations that would be appropriate to preserve creative intent on displays with colour volumes different from that of the described mastering display.

The information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of SMPTE ST 2086 (2018).

When a mastering display colour volume SEI message is present for any picture of a CLVS of a particular layer, a mastering display colour volume SEI message shall be present for the first picture of the CLVS. The mastering display colour volume SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All mastering display colour volume SEI messages that apply to the same CLVS shall have the same content.

display_primaries_x[c], when in the range of 5 to 37 000, inclusive, specifies the normalized x chromaticity coordinate of the colour primary component c of the mastering display, according to the CIE 1931 definition of x as specified in ISO

11664-1 (see also ISO 11664-3 and CIE 15), in increments of 0.00002. When display_primaries_x[c] is not in the range of 5 to 37 000, inclusive, the normalized x chromaticity coordinate of the colour primary component c of the mastering display is unknown or unspecified or specified by other means not specified in this Specification.

display_primaries_y[c], when in the range of 5 to 42 000, inclusive, specifies the normalized y chromaticity coordinate of the colour primary component c of the mastering display, according to the CIE 1931 definition of y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in increments of 0.00002. When display_primaries_y[c] is not in the range of 5 to 42 000, inclusive, the normalized y chromaticity coordinate of the colour primary component c of the mastering display is unknown or unspecified or specified by other means not specified in this Specification.

For describing mastering displays that use red, green, and blue colour primaries, it is suggested that index value c equal to 0 should correspond to the green primary, c equal to 1 should correspond to the blue primary, and c equal to 2 should correspond to the red colour primary (see also Annex E and Table E.3).

white_point_x, when in the range of 5 to 37 000, inclusive, specifies the normalized x chromaticity coordinate of the white point of the mastering display, according to the CIE 1931 definition of x as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. When white_point_x is not in the range of 5 to 37 000, inclusive, the normalized x chromaticity coordinate of the white point of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

white_point_y, when in the range of 5 to 42 000, inclusive, specifies the normalized y chromaticity coordinate of the white point of the mastering display, according to the CIE 1931 definition of y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. When white_point_y is not in the range of 5 to 42 000, inclusive, the normalized y chromaticity coordinate of the white point of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

NOTE 1 – SMPTE ST 2086 (2018) specifies that the normalized x and y chromaticity coordinate values for the mastering display colour primaries and white point are to be represented with four decimal places. This would correspond with using values of the syntax elements display_primaries_x[c], display_primaries_y[c], white_point_x, and white_point_y, as defined in this Specification, that are multiples of 5.

NOTE 2 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G (2016) uses normalized (x, y) chromaticity coordinate values of (0,0) for the white point to indicate that the white point chromaticity is unknown.

max_display_mastering_luminance, when in the range of 50 000 to 100 000 000, specifies the nominal maximum display luminance of the mastering display in units of 0.0001 candelas per square metre. When max_display_mastering_luminance is not in the range of 50 000 to 100 000 000, the nominal maximum display luminance of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

NOTE 3 – SMPTE ST 2086 (2018) specifies that the nominal maximum display luminance of the mastering display is to be specified as a multiple of 1 candela per square meter. This would correspond with using values of the syntax element max_display_mastering_luminance, as defined in this Specification, that are a multiple of 10 000.

NOTE 4 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G (2016) uses the value 0 for the nominal maximum display luminance of the mastering display to indicate that the nominal maximum display luminance of the mastering display is unknown.

min_display_mastering_luminance, when in the range of 1 to 50 000, specifies the nominal minimum display luminance of the mastering display in units of 0.0001 candelas per square metre. When min_display_mastering_luminance is not in the range of 1 to 50 000, the nominal maximum display luminance of the mastering display is unknown or unspecified or specified by other means not specified in this Specification. When max_display_mastering_luminance is equal to 50 000, min_display_mastering_luminance shall not be equal to 50 000.

NOTE 5 – SMPTE ST 2086 (2018) specifies that the nominal minimum display luminance of the mastering display is to be specified as a multiple of 0.0001 candelas per square metre, which corresponds to the semantics specified in this Specification.

NOTE 6 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G (2016) uses the value 0 for the nominal minimum display luminance of the mastering display to indicate that the nominal minimum display luminance of the mastering display is unknown.

NOTE 7 – Another example of the potential use of values outside the range for which semantics are specified in this Specification is that SMPTE ST 2086 (2018) indicates that values outside the specified range could be used to indicate that the black level and contrast of the mastering display have been adjusted using picture line-up generation equipment (PLUGE).

At the minimum luminance, the mastering display is considered to have the same nominal chromaticity as the white point.

D.3.29 Segmented rectangular frame packing arrangement SEI message semantics

This SEI message informs the decoder that the output cropped decoded picture contains samples of multiple distinct spatially packed constituent frames that are packed into one frame using a rectangular region frame packing arrangement. This information can be used by the decoder to appropriately rearrange the samples and process the samples of the constituent frames appropriately for display or other purposes (which are outside the scope of this Specification).

Each colour component plane of the output cropped decoded pictures contains a rectangular region frame packing arrangement of corresponding planes of two constituent frames as illustrated in Figure D.10.

NOTE 1 – Figure D.10 provides an illustration of the rearrangement process for the rectangular region frame packing arrangement.

This SEI message may be associated with pictures that are either frames (when field_seq_flag is equal to 0) or fields (when field_seq_flag is equal to 1). The rectangular region frame packing arrangement of the samples is specified in terms of the sampling structure of a frame in order to define a frame packing arrangement structure that is invariant with respect to whether a picture is a single field of such a packed frame or is a complete packed frame.

When general_non_packed_constraint_flag is equal to 1 in the active SPS for the current layer, there shall be no segmented rectangular frame packing arrangement SEI messages applicable for any picture of the CLVS of the current layer.

When a frame packing arrangement SEI message is applicable for any picture of the CLVS of the current layer, there shall be no segmented rectangular frame packing arrangement SEI messages applicable for any picture of the CLVS of the current layer.

segmented_rect_frame_packing_arrangement_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous segmented rectangular frame packing arrangement SEI message in output order. segmented_rect_frame_packing_arrangement_cancel_flag equal to 0 indicates that rectangular region frame packing arrangement information follows.

segmented_rect_content_interpretation_type indicates the intended interpretation of the constituent frames as specified in Table D.12. Values of segmented_rect_content_interpretation_type that do not appear in Table D.12 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore rectangular region frame packing arrangement SEI messages that contain reserved values of segmented_rect_content_interpretation_type.

For the specified frame packing arrangement scheme, there are two constituent frames that are referred to as frame 0 and frame 1.

Table D.12 – Definition of segmented_rect_content_interpretation_type

Value	Interpretation
0	Unspecified relationship between the frame packed constituent frames
1	Indicates that the two constituent frames form the left and right views of a stereo view scene, with frame 0 being associated with the left view and frame 1 being associated with the right view
2	Indicates that the two constituent frames form the right and left views of a stereo view scene, with frame 0 being associated with the right view and frame 1 being associated with the left view

NOTE 2 – The value 2 for segmented_rect_content_interpretation_type is not expected to be prevalent in industry use of this SEI message. However, the value was specified herein for purposes of completeness.

segmented_rect_frame_packing_arrangement_persistence_flag specifies the persistence of the segmented rectangular frame packing arrangement SEI message for the current layer.

segmented_rect_frame_packing_arrangement_persistence_flag equal to 0 specifies that the rectangular region frame packing arrangement SEI message applies to the current decoded frame only.

Let picA be the current picture. segmented_rect_frame_packing_arrangement_persistence_flag equal to 1 specifies that the segmented rectangular frame packing arrangement SEI message persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A frame picB in the current layer in an access unit containing a segmented rectangular frame packing arrangement SEI message applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

NOTE 3 – The default display window parameters in the VUI parameters of the SPS should be used by an encoder to indicate to a decoder that does not interpret the rectangular region frame packing arrangement SEI message that the default display window is an area within only one of the two constituent frames.

Let croppedWidth and croppedHeight be the width and height, respectively, of the cropped frame output from the decoder in units of luma samples, derived as follows:

$$\text{croppedWidth} = \text{pic\_width\_in\_luma\_samples} - \text{SubWidthC} * (\text{conf\_win\_right\_offset} + \text{conf\_win\_left\_offset}) \quad \text{(D-28)}$$

$$\text{croppedHeight} = \text{pic\_height\_in\_luma\_samples} - \text{SubHeightC} * (\text{conf\_win\_bottom\_offset} + \text{conf\_win\_top\_offset}) \quad \text{(D-29)}$$

It is a requirement of bitstream conformance for the rectangular region frame packing arrangement that croppedWidth and croppedHeight shall be integer multiples of 3.

Let oneThirdWidth and oneThirdHeight be derived as follows:

$$\text{oneThirdWidth} = \text{croppedWidth} / 3 \quad \text{(D-30)}$$

$$\text{oneThirdHeight} = \text{croppedHeight} / 3 \quad \text{(D-31)}$$

The rectangular region frame packing arrangement is composed of five rectangular regions identified as R0, R1, R2, R3 and R4 as illustrated in Figure D.10.

The width and height of the region R0 are specified in units of frame luma samples as follows:

$$r0_W = 2 * \text{oneThirdWidth} \quad \text{(D-32)}$$

$$r0_H = 2 * \text{oneThirdHeight} \quad \text{(D-33)}$$

The width and height of the region R1 are specified in units of frame luma samples as follows:

$$r1_W = \text{oneThirdWidth} \quad \text{(D-34)}$$

$$r1_H = 2 * \text{oneThirdHeight} \quad \text{(D-35)}$$

The width and height of the region R2 are specified in units of frame luma samples as follows:

$$r2_W = \text{oneThirdWidth} \quad \text{(D-36)}$$

$$r2_H = \text{oneThirdHeight} \quad \text{(D-37)}$$

The width and height of the region R3 are specified in units of frame luma samples as follows:

$$r3_W = \text{oneThirdWidth} \quad \text{(D-38)}$$

$$r3_H = \text{oneThirdHeight} \quad \text{(D-39)}$$

The width and height of the region R4 are specified in units of frame luma samples as follows:

$$r4_W = \text{oneThirdWidth} \quad \text{(D-40)}$$

$$r4_H = \text{oneThirdHeight} \quad \text{(D-41)}$$

Constituent frame 0 is obtained by cropping from the decoded frames the region R0 and constituent frame 1 is obtained by stacking vertically the regions R2 and R3 and placing the resulting rectangle to the right of the region R1. The region R4 is not part of either constituent frame and is discarded.

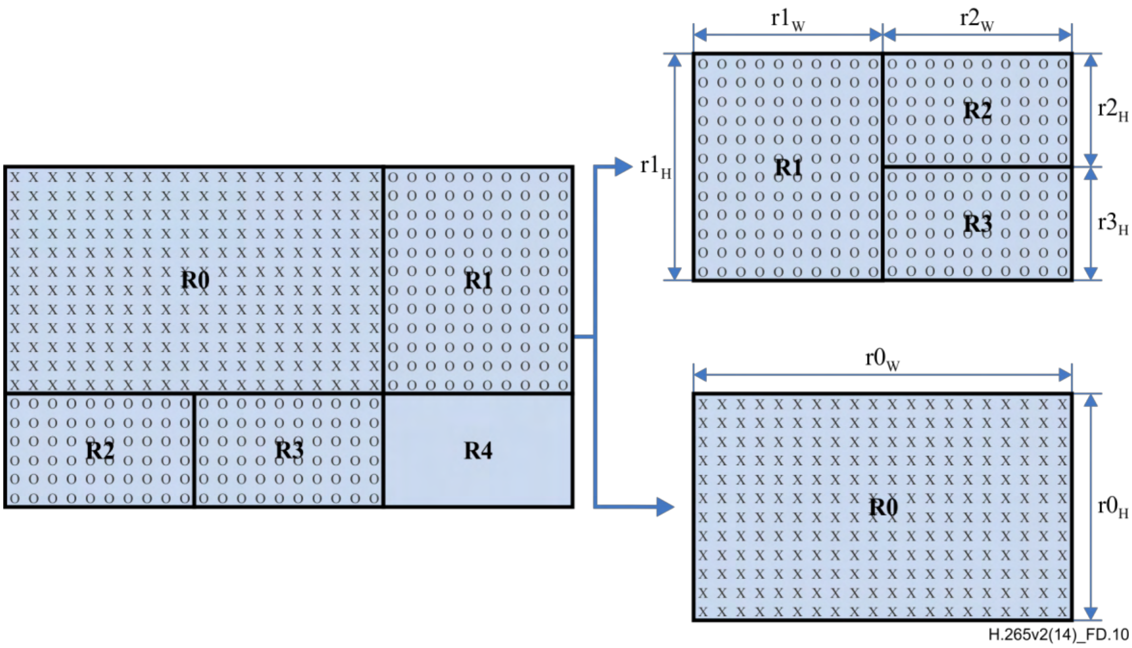


Figure D.10 – Rearrangement of a segmented rectangular frame packing arrangement

D.3.30 Temporal motion-constrained tile sets SEI message semantics

The temporal motion-constrained tile sets SEI message indicates that the following constraints apply:

– No sample values outside each identified tile set or outside the picture are referenced for inter prediction.

– For PUs located directly left of the right tile boundary of each identified tile set except the last one at the bottom right, the following applies when CuPredMode[xPb][yPb] is equal to MODE_INTER, where (xPb, yPb) specifies the top-left sample of the corresponding luma prediction block relative to the top-left sample of the current picture:

  – With the number of spatial merging candidates numSpatialMergeCand derived as follows:

$$\text{numSpatialMergeCand} = \text{availableFlagA}_0 + \text{availableFlagA}_1 + \text{availableFlagB}_0 + \text{availableFlagB}_1 + \text{availableFlagB}_2 \quad \text{(D-42)}$$

  where availableFlagA_0, availableFlagA_1, availableFlagB_0, availableFlagB_1, and availableFlagB_2 are the output of the derivation process for spatial merging candidates specified in clause 8.5.3.2.3, the following applies:

    – If numSpatialMergeCand is equal to 0, merge_flag[xPb][yPb] is equal to 0.

    – Otherwise (numSpatialMergeCand is greater than 0), merge_idx[xPb][yPb] is in the range of 0 to numSpatialMergeCand − 1, inclusive.

  – With the number of spatial motion vector predictor candidates numSpatialMvpCand derived as follows:

```
if( availableFlagLXA )
   numSpatialMvpCand =
availableFlagLXA + ( ( mvLXA != mvLXB ) ? availableFlagLXB : 0 )
else                                                              (D-43)
   numSpatialMvpCand = availableFlagLXB
```

  where availableFlagLXA, availableFlagLXB, mvLXA, and mvLXB are the output of the derivation process for motion vector predictor candidates from neighbouring prediction unit partitions specified in clause 8.5.3.2.7, the following applies:

    – If numSpatialMvpCand is equal to 0, mvp_l0_flag[xPb][yPb] and mvp_l1_flag[xPb][yPb] are equal to 1.

    – Otherwise (numSpatialMvpCand is greater than 0), mvp_l0_flag[xPb][yPb] and mvp_l1_flag[xPb][yPb] are in the range of 0 to numSpatialMvpCand − 1, inclusive.

NOTE 1 – The first constraint restricts motion vector values to only those that refer either to full-sample locations inside each identified tile set or to fractional-sample locations that require only full-sample locations inside each identified tile set for

interpolation. The second constraint prohibits the usage of motion vector candidates for temporal motion vector prediction that are derived from blocks outside each identified tile set.

Let a set of pictures associatedPicSet be the pictures with nuh_layer_id equal to targetLayerId from the access unit containing the SEI message, inclusive, up to but not including the first of any of the following in decoding order:

– The next access unit, in decoding order, that contains a temporal motion-constrained tile sets SEI message applicable to targetLayerId.
– The next IRAP picture with NoRaslOutputFlag equal to 1, in decoding order, with nuh_layer_id equal to targetLayerId.
– The next IRAP access unit, in decoding order, with NoClrasOutputFlag equal to 1.

The scope of the temporal motion-constrained tile sets SEI message is the set of pictures associatedPicSet.

When a temporal motion-constrained tile sets SEI message is present for any picture in associatedPicSet, a temporal motion-constrained tile sets SEI message applicable to targetLayerId shall be present for the first picture of associatedPicSet in decoding order and may also be present for other pictures of associatedPicSet.

When a temporal motion-constrained tile sets SEI message applicable to targetLayerId is present for a picture in associatedPicSet and tiles_enabled_flag is equal to 0 for the PPS that is active for the picture, the picture contains only one tile, which forms the only MCTS, and the values of mc_all_tiles_exact_sample_value_match_flag and each_tile_one_tile_set_flag shall both be equal to 1.

The temporal motion-constrained tile sets SEI message applicable to targetLayerId shall not be present for any picture in associatedPicSet unless every PPS that is active for any picture in associatedPicSet has the same values of the syntax elements num_tile_columns_minus1, num_tile_rows_minus1, uniform_spacing_flag, column_width_minus1[i] and row_height_minus1[i].

NOTE 2 – This constraint is similar to the constraint associated with tiles_fixed_structure_flag equal to 1 and it may be desirable for tiles_fixed_structure_flag to be equal to 1 when the temporal motion-constrained tile sets SEI message is present (although this is not required).

NOTE 3 – When loop filtering is applied across tile boundaries, inter prediction of any samples within a temporal motion-constrained tile set that refers to samples within four samples from a temporal motion-constrained tile set boundary that is not also a picture boundary may result in propagation of mismatch error. An encoder can avoid such potential error propagation by avoiding the use of motion vectors that cause such references.

When more than one temporal motion-constrained tile sets SEI message applicable to targetLayerId is present for the pictures of associatedPicSet, they shall contain identical content.

When a temporal motion-constrained tile sets SEI message is present, a slice segment that contains one or more tiles in any particular temporal motion-constrained tile set shall not be a dependent slice segment of an independent slice segment that contains one or more tiles that do not belong to that temporal motion-constrained tile set.

For purposes of referencing a particular temporal motion-constrained tile set that is identified in a temporal motion-constrained tile sets SEI message (e.g., for use with a motion-constrained tile sets extraction information sets SEI message or a motion-constrained tile sets extraction information nesting SEI message), an MCTS index is defined as follows:

– If the value of each_tile_one_tile_set_flag of the temporal motion-constrained tile sets SEI message is equal to 0, the MCTS index is the value of the variable i within the loop of the num_sets_in_message_minus1 + 1 sets of MCTS information specified by the temporal MCTS SEI message.
– Otherwise, the MCTS index of each MCTS is the tile position of the tile in tile raster scan order.

mc_all_tiles_exact_sample_value_match_flag equal to 0 indicates that when the CTUs that do not belong to any tile in the motion-constrained tile sets are not decoded and the boundaries of the tiles in the motion-contrained tile sets are treated as picture boundaries for purposes of the decoding process, the decoded value of each sample in the tiles in the motion-constrained tile sets may not be exactly the same as the decoded value of the same sample when all the CTUs of the picture are decoded. mc_all_tiles_exact_sample_value_match_flag equal to 1 indicates that when the CTUs that do not belong to any tile in the motion-constrained tile sets are not decoded and the boundaries of the tiles in the motion-constrained tile sets are treated as picture boundaries for purposes of the decoding process, the decoded value of each sample in the tiles in the motion-constrained tile sets is exactly the same as the decoded value of the sample that would be obtained when all the CTUs of all pictures in associatedPicSet are decoded.

each_tile_one_tile_set_flag equal to 1 indicates that each and every tile in the picture is included in a separate temporal motion-constrained tile set. each_tile_one_tile_set_flag equal to 0 indicates that such constraint may not be applied.

limited_tile_set_display_flag equal to 1 specifies that the display_tile_set_flag[i] syntax element is present and indicates that the tiles not included within any tile set with display_tile_set_flag[i] equal to 1 are not intended for display. limited_tile_set_display_flag equal to 0 specifies that the display_tile_set_flag[i] syntax element is not present.

num_sets_in_message_minus1 plus 1 specifies the number of temporal motion-constrained tile sets identified in the SEI message. The value of num_sets_in_message_minus1 shall be in the range of 0 to 255, inclusive.

mcts_id[i] contains an identifying number that may be used to identify the purpose of the i-th identified tile set (for example, to identify an area to be extracted from associatedPicSet for a particular purpose). The value of mcts_id[i] shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of mcts_id[i] from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of mcts_id[i] from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC and bitstreams shall not contain such values. Decoders encountering a value of mcts_id[i] in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

display_tile_set_flag[i] equal to 1 indicates that the i-th tile set is intended for display. display_tile_set_flag[i] equal to 0 indicates that the i-th tile set is not intended for display. When not present, the value of display_tile_set_flag[i] is inferred to be equal to 1.

num_tile_rects_in_set_minus1[i] plus 1 specifies the number of rectangular regions of tiles in the i-th identified temporal motion-constrained tile set. The value of num_tile_rects_in_set_minus1[i] shall be in the range of 0 to (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1, inclusive.

top_left_tile_idx[i][j] and **bottom_right_tile_idx**[i][j] identify the tile position of the top-left tile and the tile position of the bottom-right tile in a rectangular region of the i-th identified temporal motion-constrained tile set, respectively, in tile raster scan order.

The value of top_left_tile_idx[i][j] and bottom_right_tile_idx[i][j] shall be in the range of 0 to (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1, inclusive.

mc_exact_sample_value_match_flag[i] equal to 0 indicates that when the CTUs that are outside of the i-th identified temporal motion-constrained tile set are not decoded and the boundaries of the temporal motion-constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the identified tile set may not be exactly the same as the value of the same sample when all the CTUs of the picture are decoded. mc_exact_sample_value_match_flag[i] equal to 1 indicates that when the CTUs that do not belong to the temporal motion-constrained tile set are not decoded and the boundaries of the temporal motion-constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the temporal motion-constrained tile set would be exactly the same as the value of the sample that would be obtained when all the CTUs of all pictures in associatedPicSet are decoded.

NOTE 4 – It should be feasible to use mc_exact_sample_value_match_flag[i] equal to 1 when using certain combinations of loop_filter_across_tiles_enabled_flag, pps_loop_filter_across_slices_enabled_flag, pps_deblocking_filter_disabled_flag, slice_loop_filter_across_slices_enabled_flag, slice_deblocking_filter_disabled_flag, sample_adaptive_offset_enabled_flag, slice_sao_luma_flag and slice_sao_chroma_flag.

mcts_tier_level_idc_present_flag[i] equal to 1 specifies that the mcts_tier_flag[i] and mcts_level_idc[i] syntax elements are present. mcts_tier_level_idc_present_flag[i] equal to 0 specifies that the mcts_tier_flag[i] and mcts_level_idc[i] syntax elements are not present. When mcts_tier_level_idc_present_flag[i] is equal to 1, num_tile_rects_in_set_minus1[i] shall be equal to 0 and a slice segment containing one or more CTUs within the i-th motion-constrained tile set shall not include any CTU outside the i-th motion-constrained tile set. When not present, mcts_tier_level_idc_present_flag[i] is inferred to be equal to 0.

mcts_tier_flag[i] specifies the tier context for the interpretation of mcts_level_idc[i] corresponding to the i-th motion-constrained tile set. mcts_tier_flag[i] equal to 0 indicates conformance to the Main tier, and mcts_tier_flag[i] equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8. mcts_tier_flag[i] shall be equal to 0 when mcts_level_idc[i] is less than 120. When not present, the value of mcts_tier_flag[i] is inferred to be equal to general_tier_flag.

mcts_level_idc[i] indicates a level to which the i-th tile set region, corresponding to the i-th motion-constrained tile set, conforms. The value of mcts_level_idc[i] shall be less than or equal to the value of general_level_idc in the active SPS RBSP. When not present, the value of mcts_level_idc[i] is inferred to be equal to general_level_idc.

mcts_max_tier_level_idc_present_flag equal to 1 specifies that the mcts_max_tier_flag and mcts_max_level_idc syntax elements are present. mcts_max_tier_level_idc_present_flag equal to 0 specifies that the mcts_max_tier_flag and mcts_max_level_idc syntax elements are not present. When mcts_max_tier_level_idc_present_flag is equal to 1, a slice segment containing one or more CTUs within a tile shall not include any CTU outside the tile. When not present, mcts_max_tier_level_idc_present_flag is inferred to be equal to 0.

mcts_max_tier_flag specifies the tier context for the interpretation of mcts_max_level_idc to which all motion-constrained tile sets conform. mcts_max_tier_flag equal to 0 indicates conformance to the Main tier, and mcts_max_tier_flag equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8. mcts_max_tier_flag shall be equal to 0 when mcts_max_level_idc is less than 120. When not present, the value of mcts_max_tier_flag is inferred to be equal to general_tier_flag.

mcts_max_level_idc indicates a level to which all motion-constrained tile sets conform. The value of mcts_max_level_idc shall be less than or equal to the value of general_level_idc in the active SPS RBSP. When not present, the value of mcts_max_level_idc is inferred to be equal to general_level_idc.

The following describes the tier and level restrictions on the bitstreams of each motion-constrained tile set:

- If mcts_tier_level_idc_present_flag[i] and mcts_max_tier_level_idc_present_flag are both equal to 0, the mctsLevelIdc[i] of all motion-constrained tile sets are inferred to be equal to general_level_idc and the mctsTierFlag[i] of all motion-constrained tile sets are inferred to be equal to general_tier_flag and the specifications of Annex A apply to all motion-constrained tile sets.
- Otherwise (mcts_tier_level_idc_present_flag[i] or mcts_max_tier_level_idc_present_flag is equal to 1), the following applies:
  - The variables mctsLevelIdc[i], mctsTierFlag[i], mctsWidthInSamplesY[i], mctsHeightInSamplesY[i], NumTileColumnsInMCTS[i] and NumTileRowsInMCTS[i] are derived as follows:
    - If each_tile_one_tile_set_flag is equal to 0, for each tile set with index i in the range of 0 to num_sets_in_message_minus1, inclusive, the following applies:
      - mctsLevelIdc[i] is set equal to mcts_level_idc[i].
      - mctsTierFlag[i] is set equal to mcts_tier_flag[i].
      - mctsWidthInSamplesY[i] is set equal to the sum of ColumnWidthInLumaSamples[k], for all k in the range of (top_left_tile_idx[i][0] % (num_tile_rows_minus1 + 1)) to (bottom_right_tile_idx[i][0] % (num_tile_rows_minus1 + 1)), inclusive.
      - mctsHeightInSamplesY[i] is set equal to the sum of RowHeightInLumaSamples[k], for all k in the range of (top_left_tile_idx[i][0] / (num_tile_rows_minus1 + 1)) to (bottom_right_tile_idx[i][0] / (num_tile_rows_minus1 + 1)), inclusive.
      - NumTileColumnsInMCTS[i] is set equal to (bottom_right_tile_idx[i][0] % (num_tile_rows_minus1 + 1)) − (top_left_tile_idx[i][0] % (num_tile_rows_minus1 + 1)) + 1.
      - NumTileRowsInMCTS[i] is set equal to (bottom_right_tile_idx[i][0] / (num_tile_rows_minus1 + 1)) − (top_left_tile_idx[i][0] / (num_tile_rows_minus1 + 1)) + 1.
    - Otherwise (each_tile_one_tile_set_flag is equal to 1), for each tile with TileId i, with i in the range of 0 to (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1, inclusive, the following applies:
      - mctsLevelIdc[i] is set equal to mcts_max_level_idc.
      - mctsTierFlag[i] is set equal to mcts_max_tier_flag.
      - mctsWidthInSamplesY[i] is set equal to ColumnWidthInLumaSamples[i % (num_tile_rows_minus1 + 1)].
      - mctsHeightInSamplesY[i] is set equal to RowHeightInLumaSamples[i / (num_tile_rows_minus1 + 1)].
      - NumTileColumnsInMCTS[i] is set equal to 1.
      - NumTileRowsInMCTS[i] is set equal to 1.
  - The variables mctsSizeInSamplesY[i] and NumSliceSegmentsInMCTS[i] are derived as follows:
    - mctsSizeInSamplesY[i] is set equal to mctsWidthInSamplesY[i] * mctsHeightInSamplesY[i].
    - NumSliceSegmentsInMCTS[i] is set equal to the number of slice segments in the i-th motion-constrained tile set.
  - The variables mctsMaxLumaPs[i], mctsMaxCPB[i] mctsMaxSliceSegments[i], mctsMaxTileRows[i], mctsMaxTileCols[i], mctsMaxBR[i] and mctsMinCr[i] are set equal MaxLumaPs, MaxCPB, MaxSliceSegmentsPerPicture, MaxTileRows, MaxTileCols and MaxBR, respectively, specified in Table A.8 for the level indicated by mctsLevelIdc[i] with the tier indicated by mctsTierFlag[i].
  - The variable mctsMinCr[i] is set equal to MinCr specified in Table A.9 for the level indicated by mctsLevelIdc[i] with the tier indicated by mctsTierFlag[i].
  - mctsLevelIdc[i] and mctsTierFlag[i] indicate the level and tier to which the i-th motion-constrained tile set conforms, as specified in Annex A with the following modifications and additions:
    - The variable PicSizeInSamplesY is replaced with the variable mctsSizeInSamplesY[i].
    - The value of num_tile_columns_minus1 is replaced with the variable NumTileColumnsInMCTS[i] − 1.
    - The value of num_tile_rows_minus1 is replaced with the variable NumTileRowsInMCTS[i] − 1.

– The number of slice segments for the i-th motion-constrained tile set is set to NumSliceSegmentsInMCTS[i].

– The variable MaxTileCols, MaxTileRows, MaxSliceSegmentsPerPicture, MaxLumaPs, MaxBR and MinCr are replaced with the variables mctsMaxTileCols[i], mctsMaxTileRows[i], mctsMaxSliceSegments[i], mctsMaxLumaPs[i], mctsMaxBR[i] and mctsMinCr[i], respectively.

– The nominal removal time of motion-constrained tile set from the CPB shall be the same as removal time of corresponding access unit.

– The difference between consecutive output times of motion-constrained tile sets from the DPB shall be same as difference between the output times of corresponding access unit.

– The value of NumBytesInNalUnit is replaced with the sum of the sizes of all NAL units belonging to the i-th motion-constrained tile set in bytes.

D.3.31 Chroma resampling filter hint SEI message semantics

The chroma resampling filter hint SEI message signals one downsampling process and one upsampling process for the chroma components of decoded pictures. When the sampling processes signalled in the chroma resampling filter hint SEI message are used, for any number of upsampling and downsampling iterations performed on the decoded pictures, the degradation of the colour components is expected to be minimized.

The chroma resampling filter hint SEI message shall not be present in a CLVS that has chroma_format_idc equal to 0.

It is a requirement of bitstream conformance that when a chroma resampling filter hint SEI message is present in a CLVS, chroma_sample_loc_type_top_field shall be equal to chroma_sample_loc_type_bottom_field in the same CLVS.

All chroma resampling filter hint SEI messages that apply to the same CLVS shall have the same content.

ver_chroma_filter_idc identifies the vertical components of the downsampling and upsampling sets of filters as specified in Table D.13. Based on the value of ver_chroma_filter_idc, the values of verFilterCoeff[ ][ ] are derived from Table D.18. The value of ver_chroma_filter_idc shall be in the range of 0 to 2, inclusive. Values of ver_chroma_filter_idc greater than 2 are reserved for future use by ITU-T | ISO/IEC.

When ver_chroma_filter_idc is equal to 0, the chroma resampling filter in the vertical direction is unspecified.

When chroma_format_idc is equal to 1, ver_chroma_filter_idc shall be equal to 1 or 2.

Table D.13 – ver_chroma_filter_idc values

Value	Description
0	Unspecified
1	Filters signalled by ver_filter_coeff[ ][ ]
2	Filters as described in SMPTE RP 2050-1 (2012)
>2	Reserved

hor_chroma_filter_idc identifies the horizontal components of the downsampling and upsampling sets of filters as specified in Table D.14. Based on the value of hor_chroma_filter_idc, the values of horFilterCoeff[ ][ ] are derived from Table D.19. The value of hor_chroma_filter_idc shall be in the range of 0 to 2, inclusive. Values of hor_chroma_filter_idc greater than 2 are reserved for future use by ITU-T | ISO/IEC.

When hor_chroma_filter_idc is equal to 0, the chroma resampling filter in the horizontal direction is unspecified.

When chroma_format_idc is equal to 3, hor_chroma_filter_idc shall be equal to 1 or 2.

When chroma_format_idc is equal to 2 and ver_chroma_filter_idc is equal to 2, hor_chroma_filter_idc shall be equal to 0.

It is a requirement of bitstream conformance that ver_chroma_filter_idc and hor_chroma_filter_idc shall not be both equal to 0.

Table D.14 – hor_chroma_filter_idc values

Value	Description
0	Unspecified
1	Filters signalled by hor_filter_coeff[ ][ ]
2	Filters as described in the 5/3 filter description of Rec. ITU-T T.800 | ISO/IEC 15444-1
>2	Reserved

ver_filtering_field_processing_flag indicates whether the vertical operations of the downsampling and the upsampling sets of filters should be applied on a field-basis or a frame-basis. When field_seq_flag is equal to 1 and pic_struct is not within the range of 9 to 12 inclusive, ver_filtering_field_processing_flag shall be equal to 1.

Based on the values of ver_filtering_field_processing_flag and field_seq_flag, the following applies:

– If ver_filtering_field_processing_flag is equal to 1:
  – If field_seq_flag is equal to 1, each output field should be filtered independently.
  – Otherwise (field_seq_flag is equal to 0), for each decoded frame the filtering process should be applied to chroma samples belonging to the top field and to chroma samples belonging to the bottom field separately.
– Otherwise (ver_filtering_field_processing_flag is equal to 0):
  – If field_seq_flag is equal to 1, the filtering process should be performed on a frame basis. The input of the filtering process should be frames resulting from the interleaving of fields in accordance with the information conveyed by the pic_struct values.
  – Otherwise (field_seq_flag is equal to 0), the filtering process should be performed on a frame basis.

The variable chromaSampleLocType is derived as follows:

– If chroma_format_idc is equal to 1, chromaSampleLocType is set equal to chroma_sample_loc_type_top_field.
– Otherwise (chroma_format_idc is not equal to 1), chromaSampleLocType is set equal to 0.

When chromaSampleLocType is greater than 1, ver_chroma_filter_idc shall not be equal to 2.

When chromaSampleLocType is equal to 1, 3 or 5, hor_chroma_filter_idc shall not be equal to 2.

target_format_idc indicates the output sampling format of the chroma components (i.e., the position of chroma samples relative to that of the luma samples) after filtering, as specified in Table D.15. The value of target_format_idc shall be in the range of 1 to 3, inclusive. The value of target_format_idc shall not be equal to the value of chroma_format_idc.

When not present, the value of target_format_idc is inferred as follows:

– If chroma_format_idc is equal to 1 and ver_chroma_filter_idc is equal to 2, the following applies:
  – If hor_chroma_filter_idc is not equal to 2, the value of target_format_idc is inferred to be equal to 2.
  – Otherwise (hor_chroma_filter_idc is equal to 2), the value of target_format_idc is inferred to be equal to 3.
– Otherwise, if chroma_format_idc is equal to 2, the following applies:
  – If ver_chroma_filter_idc is equal to 2, the value of target_format_idc is inferred to be equal to 1.
  – Otherwise (ver_chroma_filter_idc is not equal to 2), the value of target_format_idc is inferred to be equal to 3.
– Otherwise, if chroma_format_idc is equal to 3, the following applies:
  – If ver_chroma_filter_idc is equal to 2, the value of target_format_idc is inferred to be equal to 1.
  – Otherwise (ver_chroma_filter_idc is not equal to 2), the value of target_format_idc is inferred to be equal to 2.

When chroma_format_idc is greater than 1 and target_format_idc is greater than 1, ver_chroma_filter_idc shall be equal to 0.

When chroma_format_idc is less than 3 and target_format_idc is less than 3, hor_chroma_filter_idc shall be equal to 0.

Table D.15 – Chroma sampling format indicated by target_format_idc

target_format_idc	Chroma sampling format
1	4:2:0
2	4:2:2
3	4:4:4

NOTE 1 – The logic associating sampling formats to numeric values of target_format_idc here is the same as the one used to associate sampling formats to numeric values of chroma_format_idc, as described in clause 6.2.

The variable upsamplingFlag is derived as follows:

– If chroma_format_idc is greater than target_format_idc, upsamplingFlag is set equal to 0.

– Otherwise (chroma_format_idc is less than target_format_idc), upsamplingFlag is set equal to 1.

num_vertical_filters specifies the number of filters signalled for chroma downsampling and upsampling in the vertical direction. When ver_chroma_filter_idc is equal to 1, depending on the values of chromaSampleLocType and ver_filtering_field_processing_flag, the value of num_vertical_filters shall be equal to the values specified in Table D.16.

Table D.16 – Constraints on the value of num_vertical_filters

Conditions		Mandatory value of num_vertical_filters
chromaSampleLocType	ver_filtering_field_processing_flag	
0, 1	0	2
	1	3
2, 3	0	3
	1	5
4, 5	0	3
	1	5

ver_tap_length_minus1[i] plus 1 specifies the length of the i-th filter in the vertical direction. The value of ver_tap_length_minus1[i] shall be in the range of 0 to 31, inclusive.

ver_filter_coeff[i][j] specifies the value of the j-th coefficient of the i-th filter in the vertical direction. The value of ver_filter_coeff[i][j] shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive.

The variable verTapLength[ ] is derived as follows:

– If ver_chroma_filter_idc is equal to 1, verTapLength[i] is set equal to ver_tap_length_minus1[i] plus 1 for i in the range of 0..num_vertical_filters − 1.

– Otherwise (ver_chroma_filter_idc is equal to 2), the value of verTapLength[ ] is derived as specified in Table D.18.

The variable verFilterCoeff[i][j] is derived as follows:

– If ver_chroma_filter_idc is equal to 1, verFilterCoeff[i][j] is set equal to ver_filter_coeff[i][j] for i in the range of 0..num_vertical_filters − 1, inclusive, and j in the range of 0..ver_tap_length_minus1[i], inclusive.

– Otherwise (ver_chroma_filter_idc is equal to 2), the values of verFilterCoeff[ ][ ] are derived as specified in Table D.18.

num_horizontal_filters specifies the number of filters indicated for chroma downsampling and upsampling in the horizontal direction. When hor_chroma_filter_idc is equal to 1, depending on the value of chromaSampleLocType, the value of num_horizontal_filters shall be equal to the values specified in Table D.17.

Table D.17 – Constraints on the value of num_horizontal_filters

chromaSampleLocType	Mandatory value of num_horizontal_filters
0, 2, 4	3
1, 3, 5	2

hor_tap_length_minus1[i] plus 1 specifies the length of the i-th filter in the horizontal direction. The value of hor_tap_length_minus1[i] shall be in the range of 0 to 31, inclusive.

hor_filter_coeff[i][j] specifies the value of the j-th coefficient of the i-th filter in the horizontal direction. The value of hor_filter_coeff[i][j] shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive.

The variable horTapLength[ ] is derived as follows:

– If hor_chroma_filter_idc is equal to 1, horTapLength[i] is set equal to hor_tap_length_minus1[i] plus 1 for i in the range of 0 to num_horizontal_filters − 1, inclusive.

– Otherwise (hor_chroma_filter_idc is equal to 2), the values of horTapLength[ ] are derived as specified in Table D.19.

The variable horFilterCoeff[ ][ ] is derived as follows:

– If hor_chroma_filter_idc is equal to 1, horFilterCoeff[i][j] is set equal to hor_filter_coeff[i][j] for i in the range of 0 to num_horizontal_filters − 1, inclusive, and j in the range of 0 to hor_tap_length_minus1[i], inclusive.
– Otherwise (hor_chroma_filter_idc is equal to 2), the values of horFilterCoeff[ ][ ] are derived as specified in Table D.19.

Table D.18 specifies the coefficients of the sets of chroma downsampling and upsampling filters in the vertical direction when ver_chroma_filter_idc is equal to 2.

NOTE 2 – When ver_chroma_filter_idc is equal to 2, the filter coefficient values specified in Table D.18 correspond to those described in SMPTE RP 2050-1 (2012).

Table D.19 specifies the coefficients of the sets of chroma downsampling and upsampling filters in the horizontal direction when hor_chroma_filter_idc is equal to 2.

NOTE 3 – When hor_chroma_filter_idc is equal to 2, the filter coefficient values specified in Table D.19 correspond to those described in the 5/3 filter specification part of Rec. ITU-T T.800 | ISO/IEC 15444-1.

Table D.18 – Values of verFilterCoeff and verTapLength when ver_chroma_filter_idc is equal to 2

chromaSampleLocType	ver_filtering_field_processing_flag	upsamplingFlag	verFilterCoeff[ ][ ]	verTapLength[ ]
0, 1	0	0	verFilterCoeff[0][ ] = { −3, −19, 34, 500, 500, 34, −19, −3 }	verTapLength[0] = 8
		1	verFilterCoeff[1][ ] = { 19, 103, 1037, −135 }	verTapLength[1] = 4
	1	0	verFilterCoeff[0][ ] = { −8, −26, 115, 586, 409, −48, −4, 0 }	verTapLength[0] = 8
		1	verFilterCoeff[1][ ] = { 24, −41, 1169, −128 }	verTapLength[1] = 4
			verFilterCoeff[2][ ] = { −76, 783, 330, −13 }	verTapLength[2] = 4

Table D.19 – Values of horFilterCoeff and horTapLength when hor_chroma_filter_idc is equal to 2

chromaSampleLocType	upsamplingFlag	horFilterCoeff[ ][ ]	horTapLength[ ]
0, 2, 4	0	horFilterCoeff[0][ ] = { −1, 2, 6, 2, −1 }	horTapLength[0] = 5
	1	horFilterCoeff[1][ ] = { 1 }	horTapLength[1] = 1
		horFilterCoeff[2][ ] = { 1, 1 }	horTapLength[2] = 2

The chroma resampling filtering process is applied as follows:

– The variable bottomFlag is derived as follows:
  – If field_seq_flag is equal to 1 and pic_struct is equal to 2, 10 or 12, bottomFlag is set equal to 1.
  – Otherwise, if field_seq_flag is equal to 0 and ver_filtering_field_processing_flag is equal to 1 and the output field being processed is a bottom field, bottomFlag is set equal to 1.
  – Otherwise, bottomFlag is set equal to 0.
– The variables phaseOffsetUp and phaseOffsetDown are derived as follows:
  – If bottomFlag is equal to 1, phaseOffsetUp is set equal to 2 and phaseOffsetDown is set equal to 1.
  – Otherwise (bottomFlag is equal to 0), phaseOffsetUp is set equal to 0 and phaseOffsetDown is set equal to 0.
– The vertical downsampling and upsampling filter coefficients fDv[ ][ ] and fUv[ ][ ] are specified in Table D.20.
– The horizontal downsampling and upsampling filter coefficients fDh[ ][ ] and fUh[ ][ ] are specified in Table D.21.
– The vertical downsampling and upsampling filter tap lengths lenDv[ ] and lenUv[ ] are specified in Table D.20.
– The horizontal downsampling and upsampling filter tap lengths lenDh[ ] and lenUh[ ] are specified in Table D.21.
– When chroma_format_idc is equal to 1 and target_format_idc is equal to either 2 or 3, the chroma upsampling filtering process in the vertical direction is applied once on the Cb samples and once on the Cr samples of the decoded cropped output picture as follows:
  – The variables w0 and h0 are derived as follows:

    w0 = (pic_width_in_luma_samples / SubWidthC) − (conf_win_right_offset + conf_win_left_offset)
    h0 = (pic_height_in_luma_samples >> 1) − (conf_win_top_offset + conf_win_bottom_offset) (D-44)

  – Let p0X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to h0 − 1, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and p1X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to (h0 << 1) − 1, inclusive, and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the vertical chroma upsampling process.

```
divUv[ 0 ] = 0
divUv[ 1 ] = 0
for( j = 0; j < lenUv[ phaseOffsetUp ]; j++ )
    divUv[ 0 ] += fUv[ phaseOffsetUp ][ j ]
for( j = 0; j < lenUv[ 1 + phaseOffsetUp ]; j++ )
    divUv[ 1 ] += fUv[ 1 + phaseOffsetUp ][ j ]
for( u = 0; u < w0; u++ )
    for( v = 0; v < ( h0 << 1 ); v++ ) {
      sum = 0
      posOffsetUp = v % 2 + phaseOffsetUp
      for( j = − ( lenUv[ posOffsetUp ] − 1 ) / 2; j <= lenUv[ posOffsetUp ] / 2; j++ ) {
        sum += p0X[ u ][ Clip3( 0, h0 − 1, ( v >> 1 ) + j ) ]
          * fUv[ posOffsetUp ][ j + ( lenUv[ posOffsetUp ] − 1 ) / 2 ]
      p1X[ u ][ v ] = ( sum + ( divUv[ v % 2 ] >> 1 ) ) / divUv[ v % 2 ]
    }                                                                    (D-45)
```

– When ver_filtering_field_process_flag is equal to 1 and field_seq_flag is equal to 0, the chroma upsampling filtering process in the vertical direction is applied to each field of the Cb or Cr chroma components of the cropped output frame picture p0X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to h0 − 1, inclusive, and X being either Cb or Cr. The following ordered steps apply:
  1. The array p0X is deinterleaved into two fields, p0XTop with height being equal to h0 >> 1 and p0XBottom with height being equal to h0 − (h0 >> 1).
  2. The chroma upsampling filtering process in the vertical direction according to Equation D-45 is applied to p0XTop and p0XBottom to derive the output of the filtering process p1XTop and p1XBottom.
  3. The arrays p1XTop and p1XBottom are interleaved to form the output array p1X of the upsampling filtering process.
– When chroma_format_idc is equal to either 1 or 2 and target_format_idc is equal to 3, the chroma upsampling filtering process in the horizontal direction is applied once on the Cb samples and once on the Cr samples of the decoded picture as follows:

– The variables w0 and h0 are derived as follows:

```
h0 = ( pic_height_in_luma_samples / SubHeightC ) − ( conf_win_top_offset + conf_win_bottom_offset )
w0 = ( pic_width_in_luma_samples >> 1 ) − ( conf_win_right_offset + conf_win_left_offset )
                                                                                    (D-46)
```

– Let p0X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to h0 − 1, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and p1X[i][j], with i in the range of 0 to (w0 << 1) − 1, inclusive, j in the range of 0 to h0 − 1, inclusive and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the horizontal chroma upsampling process.

```
divUh[ 0 ] = 0
divUh[ 1 ] = 0
for( j = 0; j < lenUh[ 0 ]; j++ )
    divUh[ 0 ] += fUh[ 0 ][ j ]
for( j = 0; j < lenUh[ 1 ]; j++ )
    divUh[ 1 ] += fUh[ 1 ][ j ]
for( v = 0; v < h0; v++ )
    for( u = 0; u < ( w0 << 1 ); u++ ) {
        sum = 0
        for( i = −( lenUh[ u % 2 ] − 1 ) / 2; i <= lenUh[ u % 2 ] / 2; i++ )
            sum += p0X[ Clip3( 0, w0 − 1, ( u >> 1 ) + i ) ][ v ] * fUh[ u % 2 ][ i + ( lenUh[ u % 2 ] − 1 ) / 2 ]
        p1X[ u ][ v ] = ( sum + ( divUh[ u % 2 ] >> 1 ) ) / divUh[ u % 2 ]
    }                                                                               (D-47)
```

– When chroma_format_idc is equal to either 3 or 2 and target_format_idc is equal to 1, the chroma downsampling filtering process in the vertical direction is applied once on the Cb samples and once on the Cr samples of the decoded picture as follows:

– The variables w0 and h0 are derived as follows:

```
w0 = ( pic_width_in_luma_samples / SubWidthC ) − ( conf_win_right_offset + conf_win_left_offset )
h0 = pic_height_in_luma_samples − ( conf_win_top_offset + conf_win_bottom_offset )     (D-48)
```

– Let p0X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to h0 − 1, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and p1X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to (h0 >> 1) − 1, inclusive, and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the vertical chroma downsampling process.

```
divDv = 0
for( j = 0; j < lenDv[ phaseOffsetDown ]; j++ )
    divDv += fDv[ phaseOffsetDown ][ j ]
for( u = 0; u < w0; u++ )
    for( v = 0; v < ( h0 >> 1 ); v++ ) {
        sum = 0
        for( j = − ( lenDv[ phaseOffsetDown ] − 1 ) / 2; j <= lenDv[ phaseOffsetDown ] / 2; j++ )
            sum += p0X[ u ][ Clip3( 0, h0 − 1, ( v << 1 ) + j ) ]
                    * fDv[ phaseOffsetDown ][ j + ( lenDv[ phaseOffsetDown ] − 1 ) / 2 ]
        p1X[ u ][ v ] = ( sum + ( divDv >> 1 ) ) / divDv
    }                                                                               (D-49)
```

– When ver_filtering_field_process_flag is equal to 1 and field_seq_flag is equal to 0, the chroma downsampling filtering process in the vertical direction is applied to each field of each of the Cb and Cr chroma components of the cropped output frame picture p0X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to h0 − 1, inclusive, and X being either Cb or Cr. The following ordered steps apply:

1. The array p0X is deinterleaved into two fields, p0XTop with height being equal to h0 >> 1 and p0XBottom with height being equal to h0 − (h0 >> 1).
2. The chroma downsampling filtering process in the vertical direction according to Equation D-49 is applied to p0XTop and p0XBottom to derive the output of the filtering process p1XTop and p1XBottom.

3. The arrays p1XTop and p1XBottom are interleaved to form the output array p1X of the downsampling filtering process.

– When chroma_format_idc is equal to 3 and target_format_idc is equal to either 1 or 2, the chroma downsampling filtering process in the horizontal direction is applied once on the Cb samples and once on the Cr samples of the decoded picture as follows:
  – The variables w0 and h0 are derived as follows:

```
h0 = ( pic_height_in_luma_samples / SubHeightC ) − ( conf_win_top_offset + conf_win_bottom_off
set )
w0 = pic_width_in_luma_samples − ( conf_win_right_offset + conf_win_left_offset )            (D-50)
```

  – Let p0X[i][j], with i in the range of 0 to w0 − 1, inclusive, j in the range of 0 to h0 − 1, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and p1X[i][j], with i in the range of 0 to (w0 >> 1) − 1, inclusive, j in the range of 0 to h0 − 1, inclusive, and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the horizontal chroma downsampling process.

```
divUh = 0
for( j = 0; j < lenDh; j++ )
    divDh += fDh[ 0 ][ j ]
for( v = 0; v < h0; v++ )
    for( u = 0; u < ( w0 >> 1 ); u++ ) {
        sum = 0
        for( i = −( lenDh − 1 ) / 2; i <= lenDh / 2; i++ )
            sum += p0X[ Clip3( 0, w0 − 1, ( u << 1 ) + i ) ][ v ] * fDh[ 0 ][ i + ( lenDh − 1 ) / 2 ]
        p1X[ u ][ v ] = ( sum + ( divDh >> 1 ) ) / divDh
    }                                                                                          (D-51)
```

Table D.20 – Usage of chroma filter in the vertical direction

chromaSampleLocType	ver_filtering_field_processing_flag	num_vertical_filters (when applicable)	upsamplingFlag	bottomFlag	Filter coefficients (fDv for downsampling or fUv for upsampling)	Filter tap length (lenDv for downsampling or lenUv for upsampling)
0, 1	0	2	0	–	fDv[0][j] = verFilterCoeff[0][j] j = 0..lenDv[0] − 1	lenDv[0] = verTapLength[0]
			1	–	fUv[0][j] = verFilterCoeff[1][j] j = 0..lenUv[0] − 1	lenUv[0] = verTapLength[1]
					fUv[1][j] = verFilterCoeff[1][verTapLength[1] − j − 1] j = 0..lenUv[1] − 1	lenUv[1] = verTapLength[1]
	1	3	0	0	fDv[0][j] = verFilterCoeff[0][j] j = 0..lenDv[0] − 1	lenDv[0] = verTapLength[0]
				1	fDv[1][j] = verFilterCoeff[0][verTapLength[0] − j − 1] j = 0..lenDv[1] − 1	lenDv[1] = verTapLength[0]
			1	0	fUv[0][j] = verFilterCoeff[1][j] j = 0..lenUv[0] − 1	lenUv[0] = verTapLength[1]

Table D.20 – Usage of chroma filter in the vertical direction

chromaSampleLocType	ver_filtering_field_processing_flag	num_vertical_filters (when applicable)	upsamplingFlag	bottomFlag	Filter coefficients (fDv for downsampling or fUv for upsampling)	Filter tap length (lenDv for downsampling or lenUv for upsampling)
					fUv[1][j] = verFilterCoeff[2][j] j = 0..lenUv[1] − 1	lenUv[1] = verTapLength[2]
				1	fUv[2][j] = verFilterCoeff[1][verTapLength[1] − j − 1] j = 0..lenUv[2] − 1	lenUv[2] = verTapLength[1]
					fUv[3][j] = verFilterCoeff[2][verTapLength[2] − j − 1] j = 0..lenUv[3] − 1	lenUv[3] = verTapLength[2]
2, 3	0	3	0	–	fDv[0][j] = verFilterCoeff[0][j] j = 0..lenDv[0] − 1	lenDv[0] = verTapLength[0]
			1	–	fUv[0][j] = verFilterCoeff[1][j] j = 0..lenUv[0] − 1	lenUv[0] = verTapLength[1]
					fUv[1][j] = verFilterCoeff[2][j] j = 0..lenUv[1] − 1	lenUv[1] = verTapLength[2]
	1	5	0	0	fDv[0][j] = verFilterCoeff[0][j] j = 0..lenDv[0] − 1	lenDv[0] = verTapLength[0]
				1	fDv[1][j] = verFilterCoeff[1][j] j = 0..lenDv[1] − 1	lenDv[1] = verTapLength[1]
			1	0	fUv[0][j] = verFilterCoeff[2][j] j = 0..lenUv[0] − 1	lenUv[0] = verTapLength[2]
					fUv[1][j] = verFilterCoeff[3][j] j = 0..lenUv[1] − 1	lenUv[1] = verTapLength[3]
				1	fUv[2][j] = verFilterCoeff[4][j] j = 0..lenUv[2] − 1	lenUv[2] = verTapLength[4]
					fUv[3][j] = verFilterCoeff[4][verTapLength[4] − j − 1] j = 0..lenUv[3] − 1	lenUv[3] = verTapLength[4]
4, 5	0	3	0	–	fDv[0][j] = verFilterCoeff[0][j] j = 0..lenDv[0] − 1	lenDv[0] = verTapLength[0]
			1	–	fUv[0][j] = verFilterCoeff[1][j] j = 0..lenUv[0] − 1	lenUv[0] = verTapLength[1]
					fUv[1][j] = verFilterCoeff[2][j] j = 0..lenUv[1] − 1	lenUv[1] = verTapLength[2]
	1	5	0	0	fDv[0][j] = verFilterCoeff[0][j] = 0..lenDv[0] − 1	lenDv[0] = verTapLength[0]
				1	fDv[1][j] = verFilterCoeff[1][j] j = 0..lenDv[1] − 1	lenDv[1] = verTapLength[1]

Table D.20 – Usage of chroma filter in the vertical direction

chromaSampleLocType	ver_filtering_field_processing_flag	num_vertical_filters (when applicable)	upsamplingFlag	bottomFlag	Filter coefficients (fDv for downsampling or fUv for upsampling)	Filter tap length (lenDv for downsampling or lenUv for upsampling)
			1	0	fUv[0][j] = verFilterCoeff[2][j] j = 0..lenUv[0] − 1	lenUv[0] = verTapLength[2]
					fUv[1][j] = verFilterCoeff[2][verTapLength[2] − j − 1] j = 0..lenUv[1] − 1	lenUv[1] = verTapLength[2]
				1	fUv[2][j] = verFilterCoeff[3][j] j = 0..lenUv[2] − 1	lenUv[2] = verTapLength[3]
					fUv[3][j] = verFilterCoeff[4][j] j = 0..lenUv[3] − 1	lenUv[3] = verTapLength[4]

Table D.21 – Usage of chroma filter in the horizontal direction

chromaSampleLocType	num_horizontal_filters (when applicable)	upsamplingFlag	Filter coefficients (fDh for downsampling or fUh for upsampling)	Filter tap length (lenDh for downsampling or lenUh for upsampling)
0,2,4	3	0	fDh[0][j] = horFilterCoeff[0][j] j = 0..lenDh − 1	lenDh = horTapLength[0]
		1	fUh[0][j] = horFilterCoeff[1][j] j = 0..lenUh[0] − 1	lenUh[0] = horTapLength[1]
			fUh[1][j] = horFilterCoeff[2][j] j = 0..lenUh[1] − 1	lenUh[1] = horTapLength[2]
1,3,5	2	0	fDh[0][j] = horFilterCoeff[0][j] j = 0..lenDh − 1	lenDh = horTapLength[0]
		1	fUh[0][j] = horFilterCoeff[1][j] j = 0..lenUh[0] − 1	lenUh[0] = horTapLength[1]
			fUh[1][j] = horFilterCoeff[1][horTapLength[1] − j − 1] j = 0..lenUh[1] − 1	lenUh[1] = horTapLength[1]

D.3.32 Knee function information SEI message semantics

The knee function information SEI message provides information to enable mapping of the colour samples of decoded pictures for customisation to particular display environments. The process uses a knee function to map the white level of sample values in the normalized linear RGB colour space to an appropriate luminance level and should be applied to each RGB component produced by the colour space conversion of a decoded image.

A knee function is a piece-wise linear function that starts with the point (0.0, 0.0), ends with the point (1.0, 1.0) and is specified by knee points in ascending order of index i. The coordinate of the i-th knee point is specified by (input_knee_point[i] ÷ 1 000, output_knee_point[i] ÷ 1 000). A knee function maps the input luminance level normalized in the range of 0.0 to 1.0 to the output luminance level normalized in the range of 0.0 to 1.0. An example of a knee function is shown in Figure D.11.

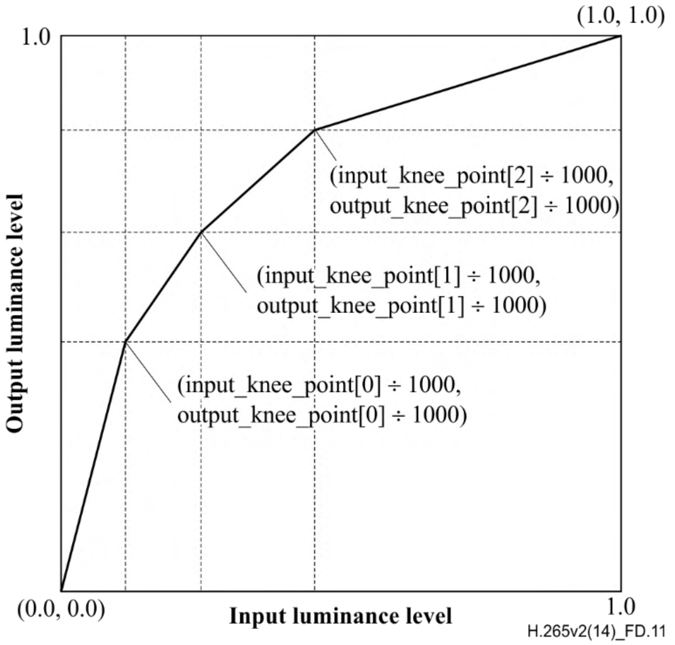


Figure D.11 – A knee function with num_knee_points_minus1 equal to 2

knee_function_id contains an identifying number that may be used to identify the purpose of the knee functions. The value of knee_function_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of knee_function_id from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of knee_function_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of knee_function_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 1 – The knee_function_id can be used to support knee function processes that are suitable for different display scenarios. For example, different values of knee_function_id may correspond to different display bit depths.

knee_function_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous knee function information SEI message in output order that applies to the current layer. knee_function_cancel_flag equal to 0 indicates that knee function information follows.

knee_function_persistence_flag specifies the persistence of the knee function information SEI message for the current layer.

knee_function_persistence_flag equal to 0 specifies that the knee function information applies to the current decoded picture only.

Let picA be the current picture. knee_function_persistence_flag equal to 1 specifies that the knee function information persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a knee function information SEI message with the same value of knee_function_id and applicable to the current layer is output for which PicOrderCnt(picB) is greater

than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

input_d_range specifies the peak luminance level for the input picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of input_d_range is equal to 0, the peak luminance level of the input picture is unspecified.

input_disp_luminance specifies the expected display brightness of peak luminance level for the input picture of the knee function process. The value of input_disp_luminance is in units of candelas per square metre. When the value of input_disp_luminance is equal to 0, the expected display brightness of peak luminance level for the input picture is unspecified.

output_d_range specifies the peak luminance level for the output picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of output_d_range is equal to 0, the peak luminance level of the output picture is unspecified.

output_disp_luminance specifies the expected display brightness of peak luminance level for the output picture of the knee function process. The value of output_disp_luminance is in units of candelas per square metre. When the value of output_disp_luminance is equal to 0, the expected display brightness of peak luminance level for the output picture is unspecified.

num_knee_points_minus1 plus 1 specifies the number of knee points used to define the knee function. num_knee_points_minus1 shall be in the range of 0 to 998, inclusive.

input_knee_point[i] specifies the luminance level of the i-th knee point of the input picture. The luminance level of the knee point of the input picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of input_knee_point[i] shall be in the range of 1 to 999, inclusive. The value of input_knee_point[i] shall be greater than the value of input_knee_point[i − 1], for i in the range of 1 to num_knee_points_minus1, inclusive.

output_knee_point[i] specifies the luminance level of the i-th knee of the output picture. The luminance level of the knee point of the output picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of output_knee_point[i] shall be in the range of 0 to 1 000, inclusive. The value of output_knee_point[i] shall be greater than or equal to the value of output_knee_point[i − 1], for i in the range of 1 to num_knee_points_minus1, inclusive.

NOTE 2 – The luminance level conversion process between an input signal x and an output signal y, where the luminance levels for both input and output are normalized to be in the range of 0.0 to 1.0, is specified as follows:

```
if( x  <=  input_knee_point[ 0 ] ÷ 1 000 )
    y = ( output_knee_point[ 0 ] ÷ input_knee_point[ 0 ] ) * x
else if( x > input_knee_point[ num_knee_points_minus1 ] )
    y = ( ( 1 000 − output_knee_point[ num_knee_points_minus1 ] ) ÷
          ( 1 000 − input_knee_point[ num_knee_points_minus1 ] ) ) *
          ( x − input_knee_point[ num_knee_points_minus1 ] ÷ 1 000 ) +
          ( output_knee_point[ num_knee_points_minus1 ] ÷ 1 000 )                    (D-52)
else
    for( i = 1; i  <=  num_knee_points_minus1; i++ )
        if( input_knee_point[ i − 1 ] ÷ 1 000 < x  &&  x  <=  input_knee_point[ i ] ÷ 1 000 )
            y = ( ( output_knee_point[ i ] − output_knee_point[ i − 1 ] ) ÷
                ( input_knee_point[ i ] − input_knee_point[ i − 1 ] ) ) *
                ( x − input_knee_point[ i − 1 ] ÷ 1 000 ) + ( output_knee_point[ i − 1 ] ÷ 1 000 )
```

D.3.33 Colour remapping information SEI message semantics

The colour remapping information SEI message provides information to enable remapping of the reconstructed colour samples of the output pictures for purposes such as converting the output pictures to a representation that is more suitable for an alternative display. The colour remapping model used in the colour remapping information SEI message is composed of a first piece-wise linear function applied to each colour component (specified by the "pre" set of syntax elements herein), followed by a three-by-three matrix applied to the three resulting colour components, followed by a second piece-wise linear function applied to each resulting colour component (specified by the "post" set of syntax elements herein).

NOTE 1 – Colour remapping of the output pictures for the display process (which is outside the scope of this Specification) is optional and does not affect the decoding process specified in this Specification.

Unless indicated otherwise by some means not specified in this Specification, the input to the indicated remapping process is the set of decoded sample values after applying an (unspecified) upsampling conversion process to the 4:4:4 colour sampling format as necessary when the colour remapping three-by-three matrix coefficients are present in the SEI message and chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format). When chroma_format_idc is equal to 0 (monochrome), the colour remapping information SEI message shall not be present, although decoders shall allow such messages to be present and shall ignore any such colour remapping information SEI messages that may be present.

colour_remap_id contains an identifying number that may be used to identify the purpose of the colour remapping information. The value of colour_remap_id may be used (in a manner not specified in this Specification) to indicate that the input to the remapping process is the output of some conversion process that is not specified in this Specification, such as a conversion of the picture to some alternative colour representation (e.g., conversion from a YCbCr colour representation to a GBR colour representation). When more than one colour remapping information SEI message is present with the same value of colour_remap_id, the content of these colour remapping information SEI messages shall be the same. When colour remapping information SEI messages are present that have more than one value of colour_remap_id, this may indicate that the remapping processes indicated by the different values of colour_remap_id are alternatives that are provided for different purposes or that a cascading of remapping processes is to be applied in a sequential order (an order that is not specified in this Specification). The value of colour_remap_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of colour_remap_id from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of colour_remap_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of colour_remap_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 2 – The colour_remap_id can be used to support different colour remapping processes that are suitable for different display scenarios. For example, different values of colour_remap_id may correspond to different remapped colour spaces supported by displays.

colour_remap_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous colour remapping information SEI message in output order that applies to the current layer. colour_remap_cancel_flag equal to 0 indicates that colour remapping information follows.

colour_remap_persistence_flag specifies the persistence of the colour remapping information SEI message for the current layer.

colour_remap_persistence_flag equal to 0 specifies that the colour remapping information applies to the current picture only.

Let picA be the current picture. colour_remap_persistence_flag equal to 1 specifies that the colour remapping information persists for the current layer in output order until either of the following conditions is true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a colour remapping information SEI message with the same value of colour_remap_id and applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

colour_remap_video_signal_info_present_flag equal to 1 specifies that syntax elements colour_remap_full_range_flag, colour_remap_primaries, colour_remap_transfer_function and colour_remap_matrix_coefficients are present, colour_remap_video_signal_info_present_flag equal to 0 specifies that syntax elements colour_remap_full_range_flag, colour_remap_primaries, colour_remap_transfer_function and colour_remap_matrix_coefficients are not present.

colour_remap_full_range_flag has the same semantics as specified in clause E.3.1 for the video_full_range_flag syntax element, except that colour_remap_full_range_flag identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_full_range_flag is inferred to be equal to the value of video_full_range_flag.

colour_remap_primaries has the same semantics as specified in clause E.3.1 for the colour_primaries syntax element, except that colour_remap_primaries identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_primaries is inferred to be equal to the value of colour_primaries.

colour_remap_transfer_function has the same semantics as specified in clause E.3.1 for the transfer_characteristics syntax element, except that colour_remap_transfer_function identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_transfer_function is inferred to be equal to the value of transfer_characteristics.

colour_remap_matrix_coefficients has the same semantics as specified in clause E.3.1 for the matrix_coeffs syntax element, except that colour_remap_matrix_coefficients identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_matrix_coefficients is inferred to be equal to the value of matrix_coeffs.

colour_remap_input_bit_depth specifies the bit depth of the colour components of the associated pictures for purposes of interpretation of the colour remapping information SEI message. When any colour remapping information SEI message is present with the value of colour_remap_input_bit_depth not equal to the bit depth of the decoded colour components,

the SEI message refers to the hypothetical result of a conversion operation performed to convert the decoded colour component samples to the bit depth equal to colour_remap_input_bit_depth.

The value of colour_remap_input_bit_depth shall be in the range of 8 to 16, inclusive. Values of colour_remap_input_bit_depth from 0 to 7, inclusive, and from 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all colour remapping SEI messages that contain a colour_remap_input_bit_depth in the range of 0 to 7, inclusive, or in the range of 17 to 255, inclusive, and bitstreams shall not contain such values.

colour_remap_output_bit_depth specifies the bit depth of the output of the colour remapping function described by the colour remapping information SEI message.

The value of colour_remap_output_bit_depth shall be in the range of 8 to 16, inclusive. Values of colour_remap_output_bit_depth from 0 to 7, inclusive, and in the range of 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all colour remapping SEI messages that contain a value of colour_remap_output_bit_depth from 0 to 7, inclusive, or in the range of 17 to 255, inclusive, and bitstreams shall not contain such values.

pre_lut_num_val_minus1[c] plus 1 specifies the number of pivot points in the piece-wise linear remapping function for the c-th component, where c equal to 0 refers to the luma or G component, c equal to 1 refers to the Cb or B component and c equal to 2 refers to the Cr or R component. When pre_lut_num_val_minus1[c] is equal to 0, the default end points of the input values are 0 and $2^{\text{colour\_remap\_input\_bit\_depth}} - 1$, and the corresponding default end points of the output values are 0 and $2^{\text{colour\_remap\_output\_bit\_depth}} - 1$, for the c-th component. In bitstreams conforming to this version of this Specification, the value of pre_lut_num_val_minus1[c] shall be in the range of 0 to 32, inclusive.

pre_lut_coded_value[c][i] specifies the input value of the i-th pivot point for the c-th component. The number of bits used to represent pre_lut_coded_value[c][i] is ((colour_remap_input_bit_depth + 7) >> 3) << 3.

pre_lut_target_value[c][i] specifies the output value of the i-th pivot point for the c-th component. The number of bits used to represent pre_lut_target_value[c][i] is ((colour_remap_output_bit_depth + 7) >> 3) << 3.

When pre_lut_coded_value[c][0] is greater than 0, an initial linear segment should be inferred that maps input values ranging from 0 to pre_lut_coded_value[c][0], inclusive, to target values ranging from 0 to pre_lut_target_value[c][0], inclusive.

When pre_lut_coded_value[c][pre_lut_num_val_minus1[c]] is not equal to $2^{\text{colour\_remap\_input\_bit\_depth}} - 1$, a final linear segment should be inferred that maps input values ranging from pre_lut_coded_value[c][pre_lut_num_val_minus1[c]] to $2^{\text{colour\_remap\_input\_bit\_depth}} - 1$, inclusive, to target values ranging from pre_lut_target_value[c][pre_lut_num_val_minus1[c]] to $2^{\text{colour\_remap\_output\_bit\_depth}} - 1$, inclusive.

colour_remap_matrix_present_flag equal to 1 indicates that the syntax elements log2_matrix_denom and colour_remap_coeffs[c][i], for c and i in the range of 0 to 2, inclusive, are present. colour_remap_matrix_present_flag equal to 0 indicates that the syntax elements log2_matrix_denom and colour_remap_coeffs[c][i], for c and i in the range of 0 to 2, inclusive, are not present.

log2_matrix_denom specifies the base 2 logarithm of the denominator for all matrix coefficients. The value of log2_matrix_denom shall be in the range of 0 to 15, inclusive. When not present, the value of log2_matrix_denom is inferred to be equal to 0.

colour_remap_coeffs[c][i] specifies the value of the three-by-three colour remapping matrix coefficients. The value of colour_remap_coeffs[c][i] shall be in the range of -2^{15} to $2^{15} - 1$, inclusive. When colour_remap_coeffs[c][i] is not present, it is inferred to be equal to 1 if c is equal to i, and inferred to be equal to 0 otherwise.

NOTE 3 – When colour_remap_matrix_present_flag is equal to 0, the colour remapping matrix is inferred to be equal to the identity matrix of size 3x3.

The variable matrixOutput[c] for c = 0, 1 and 2 is derived as follows:

```
roundingOffset = log2_matrix_denom  = =  0 ? 0 : 1  <<  ( log2_matrix_denom − 1 )
matrixOutput[ c ] = Clip3( 0, ( 1  <<  colour_remap_output_bit_depth ) − 1,
     ( colour_remap_coeffs[ c ][ 0 ] * matrixInput[ 0 ] + colour_remap_coeffs[ c ][ 1 ] * matrixInput[ 1 ]
     + colour_remap_coeffs[ c ][ 2 ] * matrixInput[ 2 ] + roundingOffset )  >>  log2_matrix_denom )
     (D-53)
```

where matrixInput[c] is the input sample value of the c-th colour component and matrixOutput[c] is the output sample value of the c-th colour component.

post_lut_num_val_minus1[c] has the same semantics as pre_lut_num_val_minus1[c], with "pre" replaced by "post", except that the default end points of the input values are 0 and $2^{\text{colour\_remap\_output\_bit\_depth}} - 1$ for the c-th colour component. The value of post_lut_num_val_minus1[c] shall be in the range of 0 to 32, inclusive.

post_lut_coded_value[c][i] has the same semantics as pre_lut_coded_value[c][i], with "pre" replaced by "post", except that the number of bits used to represent post_lut_coded_value[c][i] is ((colour_remap_output_bit_depth + 7) >> 3) << 3.

post_lut_target_value[c][i] has the same semantics as pre_lut_target_value[c][i], with "pre" replaced by "post" except that colour_remap_input_bit_depth is replaced by colour_remap_output_bit_depth in the semantics.

D.3.34 Deinterlaced field identification SEI message semantics

The deinterlaced picture information SEI message indicates that the current picture represents a frame that was interpolated via a deinterlacing process prior to encoding, and indicates the field parity of the associated source field prior to the deinterlacing process. When a progressive-to-interlace conversion process is applied to the decoded picture prior to display, it is recommended that the field of the decoded frame with the indicated field parity should be used.

When the value of field_seq_flag of any active SPS for any picture in the current access unit is equal to 1, the deinterlaced field identification SEI message shall not be present.

deinterlaced_picture_source_parity_flag equal to 0 indicates that the current picture was deinterlaced using a top field picture as the associated source field. deinterlaced_picture_source_parity_flag equal to 1 indicates that the current picture was deinterlaced using a bottom field picture as the associated source field.

D.3.35 Content light level information SEI message semantics

This SEI message identifies upper bounds for the nominal target brightness light level of the pictures of the CLVS.

The information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of the Consumer Electronics Association 861.3 specification.

The semantics of the content light level information SEI message are defined in relation to the values of samples in a 4:4:4 representation of red, green, and blue colour primary intensities in the linear light domain for the pictures of the CLVS, in units of candelas per square metre. However, this SEI message does not, by itself, identify a conversion process for converting the sample values of a decoded picture to the samples in a 4:4:4 representation of red, green, and blue colour primary intensities in the linear light domain for the picture.

NOTE 1 – Other syntax elements, such as colour_primaries, transfer_characteristics, matrix_coeffs, and the chroma resampling filter hint SEI message, when present, may assist in the identification of such a conversion process.

Given the red, green, and blue colour primary intensities in the linear light domain for the location of a luma sample in a corresponding 4:4:4 representation, denoted as E_R, E_G, and E_B, the maximum component intensity is defined as $E_{Max} = \text{Max}(E_R, \text{Max}(E_G, E_B))$. The light level corresponding to the stimulus is then defined as the CIE 1931 luminance corresponding to equal amplitudes of E_{Max} for all three colour primary intensities for red, green, and blue (with appropriate scaling to reflect the nominal luminance level associated with peak white – e.g., ordinarily scaling to associate peak white with 10 000 candelas per square metre when transfer_characteristics is equal to 16).

NOTE 2 – Since the maximum value E_{Max} is used in this definition at each sample location, rather than a direct conversion from E_R, E_G, and E_B to the corresponding CIE 1931 luminance, the CIE 1931 luminance at a location may in some cases be less than the indicated light level. This situation would occur, for example, when E_R and E_G are very small and E_B is large, in which case the indicated light level would be much larger than the true CIE 1931 luminance associated with the (E_R, E_G, E_B) triplet.

All content light level information SEI messages that apply to the same CLVS shall have the same content.

max_content_light_level, when not equal to 0, indicates an upper bound on the maximum light level among all individual samples in a 4:4:4 representation of red, green, and blue colour primary intensities (in the linear light domain) for the pictures of the CLVS, in units of candelas per square metre. When equal to 0, no such upper bound is indicated by max_content_light_level.

max_pic_average_light_level, when not equal to 0, indicates an upper bound on the maximum average light level among the samples in a 4:4:4 representation of red, green, and blue colour primary intensities (in the linear light domain) for any individual picture of the CLVS, in units of candelas per square metre. When equal to 0, no such upper bound is indicated by max_pic_average_light_level.

NOTE 3 – When the visually relevant region does not correspond to the entire cropped decoded picture, such as for "letterbox" encoding of video content with a wide picture aspect ratio within a taller cropped decoded picture, the indicated average should be performed only within the visually relevant region.

D.3.36 Dependent random access point indication SEI message semantics

The picture associated with a dependent random access point indication SEI message is referred to as a DRAP picture.

The presence of the dependent random access point indication SEI message indicates that the constraints on picture order and picture referencing specified in this clause apply. These constraints can enable a decoder to properly decode the DRAP

picture and the pictures that follow it in both decoding order and output order without needing to decode any other pictures except the associated IRAP picture.

The constraints indicated by the presence of the dependent random access point indication SEI message are as follows:

– The DRAP picture shall be a TRAIL_R picture with TemporalId equal to 0 and nuh_layer_id equal to 0.
– The DRAP picture shall not include any pictures in its RPS lists RefPicSetStCurrBefore, RefPicSetStCurrAfter, and RefPicSetLtCurr except its associated IRAP picture.
– Any picture that follows the DRAP picture in both decoding order and output order shall not include, in its RPS, any picture that precedes the DRAP picture in decoding order or output order with the exception of the IRAP picture associated with the DRAP picture.

D.3.37 Coded region completion SEI message semantics

The coded region completion SEI message indicates the value of the slice_segment_address of the next slice segment in the bitstream (when present) and whether the next slice segment in the bitstream (when present) is an independent slice segment or a dependent slice segment. The term "next slice segment", here and below in this clause, refers to the next slice segment, in decoding order, after the VCL NAL unit associated with the SEI NAL unit containing the coded region completion SEI message.

NOTE – The coded region completion SEI message may be used in determining that a complete slice, coded picture, or access unit has been received prior to receiving any VCL NAL unit of the next slice, coded picture, or access unit, respectively. Consequently, when an implementation of the decoding process expects a complete slice, coded picture, or access unit as input, the coded region completion SEI message may reduce the decoding latency.

next_segment_address identifies the value of the slice_segment_address of the next slice segment in the bitstream (when present). When the next slice segment has first_slice_segment_in_pic_flag equal to 1 or no subsequent slice segment is present in the bitstream, the value of next_segment_address shall be equal to 0.

independent_slice_segment_flag equal to 1 indicates that the next slice segment, when present, is an independent slice segment. independent_slice_segment_flag equal to 0 indicates that the next slice segment, when present, is a dependent slice segment. When independent_slice_segment_flag is not present, it is inferred to be equal to 1.

D.3.38 Alternative transfer characteristics SEI message semantics

The alternative transfer characteristics SEI message provides a preferred alternative value for the transfer_characteristics syntax element that is indicated by the colour description syntax of VUI parameters of the SPS. This SEI message is intended to be used in cases when some value of transfer_characteristics is preferred for interpretation of the pictures of the CLVS although some other value of transfer_characteristics may also be acceptable for interpretation of the pictures of the CLVS and that other value is provided in the colour description syntax of VUI parameters of the SPS for interpretation by decoders that do not support interpretation of the preferred value (e.g., because the preferred value had not yet been defined in a previous version of this Specification).

When an alternative transfer characteristics SEI message is present for any picture of a CLVS of a particular layer and the first picture of the CLVS is an IRAP picture, an alternative transfer characteristics SEI message shall be present for that IRAP picture. The alternative transfer characteristics SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All alternative transfer characteristics SEI messages that apply to the same CLVS shall have the same content.

preferred_transfer_characteristics specifies a preferred alternative value for the transfer_characteristics syntax element of the colour description syntax of VUI parameters of the SPS. The semantics for preferred_transfer_characteristics are otherwise the same as for the transfer_characteristics syntax element specified in the VUI parameters of the SPS (see clause E.3.1 and Table E.4). When preferred_transfer_characteristics is not equal to the value of transfer_characteristics indicated in the VUI parameters of the SPS, decoders should ignore the value of transfer_characteristics indicated in the VUI parameters of the SPS and instead use the value indicated by preferred_transfer_characteristics.

D.3.39 Ambient viewing environment SEI message semantics

The ambient viewing environment SEI message identifies the characteristics of the nominal ambient viewing environment for the display of the associated video content. The syntax elements of the ambient viewing environment SEI message may assist the receiving system in adapting the received video content for local display in viewing environments that may be similar or may substantially differ from those assumed or intended when mastering the video content.

This SEI message does not provide information on colour transformations that would be appropriate to preserve creative intent on displays with colour volumes different from that of the described mastering display.

When an ambient viewing environment SEI message is present for any picture of a CLVS of a particular layer and the first picture of the CLVS is an IRAP picture, an ambient viewing environment SEI message shall be present for that IRAP

picture. The ambient viewing environment SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All ambient viewing environment SEI messages that apply to the same CLVS shall have the same content.

ambient_illuminance specifies the environmental illluminance of the ambient viewing environment in units of 0.0001 lux. ambient_illuminance shall not be equal to 0.

ambient_light_x and **ambient_light_y** specify the normalized x and y chromaticity coordinates, respectively, of the environmental ambient light in the nominal viewing environment, according to the CIE 1931 definition of x and y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. The values of ambient_light_x and ambient_light_y shall be in the range of 0 to 50 000, inclusive.

NOTE – For example, the conditions identified in Rec. ITU-R BT.2035 can be expressed using ambient_illuminance equal to 100 000 with background chromaticity indicating D_{65} (ambient_light_x equal to 15 635, ambient_light_y equal to 16 450), or optionally in some regions, background chromaticity indicating D_{93} (ambient_light_x equal to 14 155, ambient_light_y equal to 14 855).

D.3.40 Content colour volume SEI message semantics

The content colour volume SEI message describes the colour volume characteristics of the associated pictures. These colour volume characteristics are expressed in terms of a nominal range, although deviations from this range may occur.

The variable transferCharacteristics is specified as follows:

– If an alternative transfer characteristics SEI message is present for the CLVS, transferCharacteristics is set equal to preferred_transfer_characteristics;

– Otherwise, (an alternative transfer characteristics SEI message is not present for the CLVS), transferCharacteristics is set equal to transfer_characteristics.

The content colour volume SEI message shall not be present, and decoders shall ignore it, when any of the following conditions is true:

– Any of the values of transferCharacteristics, colour_primaries, and matrix_coeffs has a value defined as unspecified.

– The value of transfer_characteristics is equal to 2, 4, or 5.

– The value of colour_primaries is equal to 2.

The following applies when converting the signal from a non-linear to a linear representation:

– If the value of transferCharacteristics is equal to 1, 6, 7, 14, or 15, the Rec. ITU-R BT.1886-0 reference electro-optical transfer function should be used to convert the signal to its linear representation, where the value of screen luminance for white is set equal to 100 cd/m^2, the value of screen luminance for black is set equal to 0 cd/m^2, and the value of the exponent of the power function is set equal to 2.4.

– Otherwise, if the value of transferCharacteristics is equal to 18, the hybrid log-gamma reference electro-optical transfer function specified in Rec. ITU-R BT.2100-2 should be used to convert the signal to its linear representation, where the value of nominal peak luminance of the display is set equal to 1 000 cd/m^2, the value of the display luminance for black is set equal to 0 cd/m^2, and the value of system gamma is set equal to 1.2.

– Otherwise (the value of transferCharacteristics is not equal to 1, 6, 7, 14, 15, or 18) when the content colour volume SEI message is present, the exact inverse of the transfer function specified in Table E.4 should be used to convert the non-linear signal to a linear representation.

ccv_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous content colour volume SEI message in output order that applies to the current layer. ccv_cancel_flag equal to 0 indicates that content colour volume information follows.

ccv_persistence_flag specifies the persistence of the content colour volume SEI message for the current layer.

ccv_persistence_flag equal to 0 specifies that the content colour volume applies to the current decoded picture only.

Let picA be the current picture. ccv_persistence_flag equal to 1 specifies that the content colour volume SEI message persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a content colour volume SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where

PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for the picture order count of picB.

ccv_primaries_present_flag equal to 1 specifies that the syntax elements ccv_primaries_x[c] and ccv_primaries_y[c] are present. ccv_primaries_present_flag equal to 0 specifies that the syntax elements ccv_primaries_x[c] and ccv_primaries_y[c] are not present.

ccv_min_luminance_value_present_flag equal to 1 specifies that the syntax element ccv_min_luminance_value is present. ccv_min_luminance_value_present_flag equal to 0 specifies that the syntax element ccv_min_luminance_value is not present.

ccv_max_luminance_value_present_flag equal to 1 specifies that the syntax element ccv_max_luminance_value is present. ccv_max_luminance_value_present_flag equal to 0 specifies that the syntax element ccv_max_luminance_value is not present.

ccv_avg_luminance_value_present_flag equal to 1 specifies that the syntax element ccv_avg_luminance_value is present. ccv_avg_luminance_value_present_flag equal to 0 specifies that the syntax element ccv_avg_luminance_value is not present.

It is a requirement of bitstream conformance that the values of ccv_primaries_present_flag, ccv_min_luminance_value_present_flag, ccv_max_luminance_value_present_flag, and ccv_avg_luminance_value_present_flag shall not all be equal to 0.

ccv_reserved_zero_2bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for reserved_zero_2bits[i] are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of reserved_zero_2bits[i].

ccv_primaries_x[c] and **ccv_primaries_y**[c] specify the normalized x and y chromaticity coordinates, respectively, of the colour primary component c of the nominal content colour volume, according to the CIE 1931 definition of x and y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. For describing colour volumes that use red, green, and blue colour primaries, it is suggested that index value c equal to 0 should correspond to the green primary, c equal to 1 should correspond to the blue primary, and c equal to 2 should correspond to the red colour primary (see also Annex E and Table E.3).

The values of ccv_primaries_x[c] and ccv_primaries_y[c] shall be in the range of −5 000 000 to 5 000 000, inclusive.

When ccv_primaries_x[c] and ccv_primaries_y[c] are not present, they are inferred to be equal to the normalized x and y chromaticity coordinates, respectively, specified by colour_primaries.

ccv_min_luminance_value specifies the normalized minimum luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_o or L_c as specified in Table E.4 according to the indicated transfer characteristics of the signal. The values of ccv_min_luminance_value are in normalized increments of 0.0000001.

ccv_max_luminance_value specifies the maximum luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_o or L_c as specified in Table E.4 according to the transfer characteristics of the signal. The values of ccv_max_luminance_value are in normalized increments of 0.0000001.

ccv_avg_luminance_value specifies the average luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_o or L_c as specified in Table E.4 according to the transfer characteristics of the signal. The values of ccv_avg_luminance_value are in normalized increments of 0.0000001.

NOTE – The resulting domain from this conversion process may or may not represent light in a source or display domain – it is merely a gamut representation domain rather than necessarily being a representation of actual light in either the scene or display domain. Therefore, the values corresponding to ccv_min_luminance_value, ccv_max_luminance_value, and ccv_avg_luminance_value might not necessarily correspond to a true luminance value.

The value of ccv_min_luminance_value, when present, shall be less than or equal to ccv_avg_luminance_value, when present. The value of ccv_avg_luminance_value, when present, shall be less than or equal to ccv_max_luminance_value, when present. The value of ccv_min_luminance_value, when present, shall be less than or equal to ccv_max_luminance_value, when present.

When the visually relevant region does not correspond to the entire cropped decoded picture, such as for "letterbox" encoding of video content with a wide picture aspect ratio within a taller cropped decoded picture, the indicated ccv_min_luminance_value, ccv_max_luminance_value, and ccv_avg_luminance_value should correspond only to values within the visually relevant region.

D.3.41 Semantics of omnidirectional video specific SEI messages

D.3.41.1 Equirectangular projection SEI message semantics

The equirectangular projection SEI message provides information to enable remapping (through an equirectangular projection) of the colour samples of the projected pictures onto a sphere coordinate space in sphere coordinates (ϕ, θ) for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

When an equirectangular projection SEI message is present for any picture of a CLVS of a particular layer, an equirectangular projection SEI message shall be present for the first picture of the CLVS and no SEI message indicating a different type of projection shall be present for any picture of the CLVS.

When general_non_packed_constraint_flag is equal to 1 in the active SPS for the current layer, there shall be no equirectangular projection SEI messages applicable for any picture of the CLVS of the current layer.

When aspect_ratio_idc is present and greater than 1 in the active SPS for the current layer, there should be no equirectangular projection SEI messages applicable for any picture of the CLVS of the current layer.

A frame packing arrangement SEI message for which all the following conditions are true is referred to as an effectively applicable frame packing arrangement SEI message:

– The value of frame_packing_arrangement_cancel_flag is equal to 0.

– The value of frame_packing_arrangement_type is equal to 3, 4, or 5.

– The value of quincunx_sampling_flag is equal to 0.

– The value of spatial_flipping_flag is equal to 0.

– The value of field_views_flag is equal to 0.

– The value of frame0_grid_position_x is equal to 0.

– The value of frame0_grid_position_y is equal to 0.

– The value of frame1_grid_position_x is equal to 0.

– The value of frame1_grid_position_y is equal to 0.

When a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message, an equirectangular projection SEI message with erp_cancel_flag equal to 0 that applies to the picture shall not be present. Decoders shall ignore equirectangular projection SEI messages when a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message.

When a segmented rectangular frame packing arrangement SEI message with segmented_rect_frame_packing_arrangement_cancel_flag equal to 0 is present that applies to the picture, an equirectangular projection SEI message with erp_cancel_flag equal to 0 shall not be present that applies to the picture. Decoders shall ignore equirectangular projection SEI messages when a segmented rectangular frame packing arrangement SEI message with segmented_rect_frame_packing_arrangement_cancel_flag equal to 0 is present that applies to the picture.

erp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous equirectangular projection SEI message in output order. erp_cancel_flag equal to 0 indicates that equirectangular projection information follows.

erp_persistence_flag specifies the persistence of the equirectangular projection SEI message for the current layer.

erp_persistence_flag equal to 0 specifies that the equirectangular projection SEI message applies to the current decoded picture only.

Let picA be the current picture. erp_persistence_flag equal to 1 specifies that the equirectangular projection SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing an equirectangular projection SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where

PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

erp_guard_band_flag equal to 1 indicates that the constituent picture contains guard band areas for which the sizes are specified by the syntax elements erp_left_guard_band_width and erp_right_guard_band_width. erp_guard_band_flag equal to 0 indicates that the constituent picture does not contains guard band areas for which the sizes are specified by the syntax elements erp_left_guard_band_width and erp_right_guard_band_width.

erp_reserved_zero_2bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for erp_reserved_zero_2bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of erp_reserved_zero_2bits.

erp_guard_band_type indicates the type of the guard bands as follows:

– erp_guard_band_type equal to 0 indicates that the content of the guard band in relation to the content of the constituent picture is unspecified.

– erp_guard_band_type equal to 1 indicates that the content of the guard band suffices for interpolation of sample values at sub-pel sample fractional locations within the constituent picture.

  NOTE – erp_guard_band_type equal to 1 could be used when the boundary samples of a constituent picture have been copied horizontally to the guard band.

– erp_guard_band_type equal to 2 indicates that the content of the guard band represents actual picture content at a quality that gradually changes from the picture quality of the constituent picture to that of the spherically adjacent region.

– erp_guard_band_type equal to 3 indicates that the content of the guard bands represents actual picture content at a similar level of quality as the constituent picture.

– erp_guard_band_type values greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value of erp_guard_band_type when the value is greater than 3 as equivalent to the value 0.

erp_left_guard_band_width specifies the width of the guard band on the left side of the constituent picture in units of luma samples. When erp_guard_band_flag is equal to 0, the value of erp_left_guard_band_width is inferred to be equal to 0. When chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), erp_left_guard_band_width shall be an even number.

erp_right_guard_band_width specifies the width of the guard band on the right side of the constituent picture in units of luma samples. When erp_guard_band_flag is equal to 0, the value of erp_right_guard_band_width is inferred to be equal to 0. When chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), erp_right_guard_band_width shall be an even number.

D.3.41.2 Cubemap projection SEI message semantics

The cubemap projection SEI message provides information to enable remapping (through a cubemap projection) of the colour samples of the projected pictures onto a sphere coordinate space in sphere coordinates (ϕ, θ) for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

When a cubemap projection SEI message is present for any picture of a CLVS of a particular layer, a cubemap projection SEI message shall be present for the first picture of the CLVS and no SEI message indicating a different type of projection shall be present for any picture of the CLVS.

When general_non_packed_constraint_flag is equal to 1 in the active SPS for the current layer, there shall be no cubemap projection SEI messages applicable for any picture of the CLVS of the current layer.

When aspect_ratio_idc is present and greater than 1 in the active SPS for the current layer, there should be no cubemap projection SEI messages applicable for any picture of the CLVS of the current layer.

A frame packing arrangement SEI message for which all the following conditions are true is referred to as an effectively applicable frame packing arrangement SEI message:

– The value of frame_packing_arrangement_cancel_flag is equal to 0.

– The value of frame_packing_arrangement_type is equal to 3, 4, or 5.

– The value of quincunx_sampling_flag is equal to 0.

– The value of spatial_flipping_flag is equal to 0.

– The value of field_views_flag is equal to 0.

– The value of frame0_grid_position_x is equal to 0.

– The value of frame0_grid_position_y is equal to 0.

– The value of frame1_grid_position_x is equal to 0.

– The value of frame1_grid_position_y is equal to 0.

When a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message, a cubemap projection SEI message with cmp_cancel_flag equal to 0 that applies to the picture shall not be present. Decoders shall ignore cubemap projection SEI messages when a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message.

When a segmented rectangular frame packing arrangement SEI message with segmented_rect_frame_packing_arrangement_cancel_flag equal to 0 is present that applies to the picture, a cubemap projection SEI message with cmp_cancel_flag equal to 0 shall not be present that applies to the picture. Decoders shall ignore cubemap projection SEI messages when a segmented rectangular frame packing arrangement SEI message with segmented_rect_frame_packing_arrangement_cancel_flag equal to 0 is present that applies to the picture.

cmp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous cubemap projection SEI message in output order. cmp_cancel_flag equal to 0 indicates that cubemap projection information follows.

cmp_persistence_flag specifies the persistence of the cubemap projection SEI message for the current layer.

cmp_persistence_flag equal to 0 specifies that the cubemap projection SEI message applies to the current decoded picture only.

Let picA be the current picture. cmp_persistence_flag equal to 1 specifies that the cubemap projection SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a cubemap projection SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

D.3.41.3 Fisheye video information SEI message semantics

The presence of the fisheye video information SEI message for any picture of a CLVS indicates that the picture is a fisheye video picture containing a number of active areas captured by fisheye camera lens. The information carried in the fisheye video information SEI message enables remapping of the colour samples of the pictures onto a sphere coordinate space in sphere coordinates (ϕ, θ), for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

When a fisheye video information SEI message is present for any picture of a CLVS of a particular layer, a fisheye video information SEI message shall be present for the first picture of the CLVS and no equirectangular projection SEI message or cubemap projection SEI message shall be present for any picture of the CLVS.

When general_non_packed_constraint_flag is equal to 1 in the active SPS for the current layer, there shall be no fisheye video information SEI messages applicable for any picture of the CLVS of the current layer.

When aspect_ratio_idc is present and greater than 1 in the active SPS for the current layer, there should be no fisheye video information SEI messages applicable for any picture of the CLVS of the current layer.

When a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0 or a segmented rectangular frame packing arrangement SEI message with segmented_rect_frame_packing_arrangement_cancel_flag equal to 0 that applies to the picture is present, a fisheye video information SEI message with fisheye_cancel_flag equal to 0 that applies to the picture shall not be present. Decoders shall ignore fisheye video information SEI messages when a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0 or a segmented rectangular frame packing arrangement SEI message with segmented_rect_frame_packing_arrangement_cancel_flag equal to 0 that applies to the picture is present.

fisheye_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous fisheye video information SEI message in output order. fisheye_cancel_flag equal to 0 indicates that fisheye video information follows.

fisheye_persistence_flag specifies the persistence of the fisheye video information SEI message for the current layer.

fisheye_persistence_flag equal to 0 specifies that the fisheye video information SEI message applies to the current decoded picture only.

Let picA be the current picture. fisheye_persistence_flag equal to 1 specifies that the fisheye video information SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a fisheye video information SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

fisheye_view_dimension_idc indicates the alignment and viewing direction of a fisheye lens, as follows:

– fisheye_view_dimension_idc equal to 0 indicates that fisheye_num_active_areas is equal to 2, and the values of fisheye_camera_centre_azimuth, fisheye_camera_centre_elevation, fisheye_camera_centre_tilt, fisheye_camera_centre_offset_x, fisheye_camera_centre_offset_y, and fisheye_camera_centre_offset_z are such that the active areas have aligned optical axes and face opposite directions, and the sum of fisheye_field_of_view values is greater than or equal to $360 * 2^{16}$.

– fisheye_view_dimension_idc equal to 1 indicates that fisheye_num_active_areas is equal to 2, and the values of fisheye_camera_centre_azimuth, fisheye_camera_centre_elevation, fisheye_camera_centre_tilt, fisheye_camera_centre_offset_x, fisheye_camera_centre_offset_y, and fisheye_camera_centre_offset_z are such that the active areas have parallel optical axes that are orthogonal to the line intersecting the camera centre points, and the camera corresponding to i equal to 0 is the left view.

– fisheye_view_dimension_idc equal to 2 indicates that fisheye_num_active_areas is equal to 2, and the values of fisheye_camera_centre_azimuth, fisheye_camera_centre_elevation, fisheye_camera_centre_tilt, fisheye_camera_centre_offset_x, fisheye_camera_centre_offset_y, and fisheye_camera_centre_offset_z are such that the active areas have parallel optical axes that are orthogonal to the line intersecting the camera centre points, and the camera corresponding to i equal to 0 is the right view.

– fisheye_view_dimension_idc equal to 7 indicates that no additional constraints are implied for the syntax element values within the fisheye video information SEI message.

– Values of fisheye_view_dimension_idc in the range of 3 to 6, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of fisheye_view_dimension_idc in the range of 3 to 6, inclusive, shall ignore it.

fisheye_reserved_zero_3bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for fisheye_reserved_zero_3bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of fisheye_reserved_zero_3bits.

fisheye_num_active_areas_minus1 plus 1 specifies the number of active areas in the coded picture. The value of fisheye_num_active_areas_minus1 shall be in the range of 0 to 3, inclusive. Values of fisheye_num_active_areas_minus1 greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a fisheye video information SEI message with fisheye_num_active_areas_minus1 greater than 3 shall ignore the fisheye video information SEI message.

fisheye_circular_region_centre_x[i] and **fisheye_circular_region_centre_y**[i] specify the horizontal and vertical coordinates of the centre of the circular region that contains the i-th active area in the coded picture, respectively, in units of 2^{-16} luma samples. The value of fisheye_circular_region_centre_x[i] and fisheye_circular_region_centre_y[i] shall be in the range of 0 to $65\,536 * 2^{16} - 1$ (i.e., 4 294 967 295), inclusive.

fisheye_rect_region_top[i], **fisheye_rect_region_left**[i], **fisheye_rect_region_width**[i], and **fisheye_rect_region_height**[i] specify the coordinates of the top-left corner and the width and height of the i-th rectangular region that contains the i-th active area, in units of luma samples.

The value of fisheye_rect_region_top[i] shall be in the range of SubHeightC * conf_win_top_offset to pic_height_in_luma_samples − (SubHeightC * conf_win_bottom_offset + 1), inclusive.

The value of fisheye_rect_region_left[i] shall be in the range of SubWidthC * conf_win_left_offset to pic_width_in_luma_samples − (SubWidthC * conf_win_right_offset + 1), inclusive.

The value of fisheye_rect_region_width[i] shall be in the range of 1 to pic_width_in_luma_samples − SubWidthC * (conf_win_left_offset + conf_win_right_offset), inclusive.

The value of fisheye_rect_region_height[i] shall be in the range of 1 to pic_height_in_luma_samples − SubHeightC * (conf_win_top_offset + conf_win_bottom_offset), inclusive.

The sum of fisheye_rect_region_top[i] and fisheye_rect_region_height[i] shall be less than or equal to pic_height_in_luma_samples − SubHeightC * conf_win_bottom_offset.

The sum of fisheye_rect_region_left[i] and fisheye_rect_region_width[i] shall be less than or equal to pic_width_in_luma_samples − SubWidthC * conf_win_right_offset.

fisheye_circular_region_radius[i] specifies the radius of the circular region that contains the i-th active area that is defined as a length from the centre of the circular region specified by fisheye_circular_region_centre_x[i] and fisheye_circular_region_centre_y[i] to the outermost pixel boundary of the circular region, in units of 2^{-16} luma samples, that corresponds to the maximum field of view of the i-th fisheye lens, specified by fisheye_field_of_view[i]. The value of fisheye_circular_region_radius[i] shall be in the range of 0 to 65 536 * 2^{16} − 1 (i.e., 4 294 967 295), inclusive.

The i-th active area is defined as the intersection of the i-th rectangular region, specified by fisheye_rect_region_top[i], fisheye_rect_region_left[i], fisheye_rect_region_width[i], and fisheye_rect_region_height[i], and the i-th circular region, specified by fisheye_circular_region_centre_x[i], fisheye_circular_region_centre_y[i], and fisheye_circular_region_radius[i].

Each active area shall contain at least one sample location. There shall not be any sample location that is within more than one active area.

fisheye_scene_radius[i] specifies the radius of a circular region within the i-th active area in units of 2^{-16} luma samples, where the obstruction, such as the camera body, is not included in the region specified by fisheye_circular_region_centre_x[i], fisheye_circular_region_centre_y[i], and fisheye_scene_radius[i]. The value of fisheye_scene_radius[i] shall be less than or equal to fisheye_circular_region_radius[i], and shall be in the range of 0 to 65 536 * 2^{16} − 1 (i.e., 4 294 967 295), inclusive. The enclosed area is the suggested area for stitching as recommended by the encoder.

fisheye_camera_centre_azimuth[i] and **fisheye_camera_centre_elevation**[i] indicate the sphere coordinates that correspond to the centre of the circular region that contains the i-th active area in the cropped output picture, in units of 2^{-16} degrees. The value of fisheye_camera_centre_azimuth[i] shall be in the range of −180 * 2^{16} (i.e., −11 796 480) to 180 * 2^{16} − 1 (i.e., 11 796 479), inclusive, and the value of fisheye_camera_centre_elevation[i] shall be in the range of −90 * 2^{16} (i.e., −5 898 240) to 90 * 2^{16} (i.e., 5 898 240), inclusive.

fisheye_camera_centre_tilt[i] indicates the tilt angle of the sphere region that corresponds to the i-th active area of the cropped output picture, in units of 2^{-16} degrees. The value of fisheye_camera_centre_tilt[i] shall be in the range of −180 * 2^{16} (i.e., −11 796 480) to 180 * 2^{16} − 1 (i.e., 11 796 479), inclusive.

fisheye_camera_centre_offset_x[i], **fisheye_camera_centre_offset_y**[i] and **fisheye_camera_centre_offset_z**[i] indicate the XYZ offset values, in units of 2^{-16} millimeters, of the focal centre of the fisheye camera lens corresponding to the i-th active area from the focal centre origin of the overall fisheye camera configuration. The value of each of fisheye_camera_centre_offset_x[i], fisheye_camera_centre_offset_y[i], and fisheye_camera_centre_offset_z[i] shall be in the range of 0 to 65 536 * 2^{16} − 1 (i.e., 4 294 967 295), inclusive.

fisheye_field_of_view[i] specifies the field of view of the lens that corresponds to the i-th active area in the coded picture, in units of 2^{-16} degrees. The value of fisheye_field_of_view[i] shall be in the range of 0 to 360 * 2^{16} (i.e., 23 592 960), inclusive.

fisheye_num_polynomial_coeffs[i] specifies the number of polynomial coefficients for the circular region corresponding to the i-th active area. The value of fisheye_num_polynomial_coeffs[i] shall be in the range of 0 to 8, inclusive. Values of fisheye_num_polynomial_coeffs[i] greater than 8 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a fisheye video information SEI message with fisheye_num_polynomial_coeffs[i] greater than 8 shall ignore the fisheye video information SEI message.

fisheye_polynomial_coeff[i][j] specifies the j-th polynomial coefficient value, in units of 2^{-24}, of the curve function that maps the normalized distance of a luma sample from the centre of the circular region corresponding to the i-th active area to the angular value of a sphere coordinate from the normal vector of a nominal imaging plane that passes through the centre of the sphere coordinate system for the i-th active region. The value of fisheye_polynomial_coeff[i][j] shall be in the range of −128 * 2^{24} (i.e., 2 147 483 648) to 128 * 2^{24} − 1 (i.e., 2 147 483 647), inclusive.

D.3.41.4 Sphere rotation SEI message semantics

The sphere rotation SEI message provides information on rotation angles yaw (α), pitch (β), and roll (γ) that are used for conversion between the global coordinate axes and the local coordinate axes.

Relative to an (x, y, z) Cartesian coordinate system, yaw expresses a rotation around the z (vertical, up) axis, pitch rotates around the y (lateral, side-to-side) axis, and roll rotates around the x (back-to-front) axis. Rotations are extrinsic, i.e., around x, y, and z fixed reference axes. The angles increase clockwise when looking from the origin towards the positive end of an axis.

sphere_rotation_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous sphere rotation SEI message in output order. sphere_rotation_cancel_flag equal to 0 indicates that sphere rotation information follows.

sphere_rotation_persistence_flag specifies the persistence of the sphere rotation SEI message for the current layer.

sphere_rotation_persistence_flag equal to 0 specifies that the sphere rotation SEI message applies to the current decoded picture only.

Let picA be the current picture. sphere_rotation_persistence_flag equal to 1 specifies that the sphere rotation SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a sphere rotation SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

When an equirectangular projection SEI message with erp_cancel_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 is not present in the CLVS that applies to the current picture and precedes the sphere rotation SEI message in decoding order, a sphere rotation SEI message with sphere_rotation_cancel_flag equal to 0 shall not be present in the CLVS that applies to the current picture. Decoders shall ignore sphere rotation SEI messages with sphere_rotation_cancel_flag equal to 0 that do not follow, in decoding order, an equirectangular projection SEI message with erp_cancel_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 in the CLVS that applies to the current picture.

sphere_rotation_reserved_zero_6bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for sphere_rotation_reserved_zero_6bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of sphere_rotation_reserved_zero_6bits.

yaw_rotation specifies the value of the yaw rotation angle, in units of 2^{-16} degrees. The value of yaw_rotation shall be in the range of $-180 * 2^{16}$ (i.e., −11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. When not present, the value of yaw_rotation is inferred to be equal to 0.

pitch_rotation specifies the value of the pitch rotation angle, in units of 2^{-16} degrees. The value of pitch_rotation shall be in the range of $-90 * 2^{16}$ (i.e., −5 898 240) to $90 * 2^{16}$ (i.e., 5 898 240), inclusive. When not present, the value of pitch_rotation is inferred to be equal to 0.

roll_rotation specifies the value of the roll rotation angle, in units of 2^{-16} degrees. The value of roll_rotation shall be in the range of $-180 * 2^{16}$ (i.e., −11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. When not present, the value of roll_rotation is inferred to be equal to 0.

D.3.41.5 Region-wise packing SEI message semantics

The region-wise packing SEI message provides information to enable remapping of the colour samples of the cropped decoded pictures onto projected pictures as well as information on the location and size of the guard bands, if any.

rwp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous region-wise packing SEI message in output order. rwp_cancel_flag equal to 0 indicates that region-wise packing information follows.

rwp_persistence_flag specifies the persistence of the region-wise packing SEI message for the current layer.

rwp_persistence_flag equal to 0 specifies that the region-wise packing SEI message applies to the current decoded picture only.

Let picA be the current picture. rwp_persistence_flag equal to 1 specifies that the region-wise packing SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a region-wise packing SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where

PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

When an equirectangular projection SEI message with erp_cancel_flag equal to 0 and erp_guard_band_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 is not present in the CLVS that applies to the current picture and precedes the region-wise packing SEI message in decoding order, a region-wise packing SEI message with rwp_cancel_flag equal to 0 shall not be present in the CLVS that applies to the current picture. Decoders shall ignore region-wise packing SEI messages with rwp_cancel_flag equal to 0 that do not follow, in decoding order, an equirectangular projection SEI message with erp_cancel_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 in the CLVS that applies to the current picture.

For the frame packing arrangement scheme indicated by a frame packing arrangement SEI message that applies to the current picture, if a region-wise packing SEI message with rwp_cancel_flag equal to 0 is present that applies to the current picture, the frame packing arrangement scheme applies to the projected picture, otherwise, the frame packing arrangement scheme applies to the cropped decoded picture.

If a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0, frame_packing_arrangement_type equal to 3, 4, or 5, and quincunx_sampling_flag equal to 0 is not present that applies to the current picture, the variables StereoFlag, TopBottomFlag, SideBySideFlag, and TempInterleavingFlag are all set equal to 0, the variables HorDiv1 and VerDiv1 are both set equal to 1. Otherwise the following applies:

– StereoFlag is set equal to 1.

– When the frame_packing_arrangement_type is equal to 3, SideBySideFlag is set equal to 1, TopBottomFlag and TempInterleavingFlag are both set equal to 0, HorDiv1 is set equal to 2 and VerDiv1 is set equal to 1.

– When the frame_packing_arrangement_type is equal to 4, TopBottomFlag is set equal to 1, SideBySideFlag and TempInterleavingFlag are both set equal to 0, HorDiv1 is set equal to 1 and VerDiv1 is set equal to 2.

– When the frame_packing_arrangement_type is equal to 5, TempInterleavingFlag is set equal to 1, TopBottomFlag and SideBySideFlag are both set equal to 0, HorDiv1 and VerDiv1 are both set equal to 1.

constituent_picture_matching_flag equal to 1 specifies that the projected region information, packed region information, and guard band region information in this SEI message apply individually to each constituent picture and that the packed picture and the projected picture have the same stereoscopic frame packing format indicated by the frame packing arrangement SEI message. constituent_picture_matching_flag equal to 0 specifies that the projected region information, packed region information, and guard band region information in this SEI message apply to the projected picture.

When either of the following conditions is true, the value of constituent_picture_matching_flag shall be equal to 0:

– StereoFlag is equal to 0.

– StereoFlag is equal to 1 and frame_packing_arrangement_type is equal to 5.

rwp_reserved_zero_5bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for rwp_reserved_zero_56bits[i] are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of rwp_reserved_zero_5bits[i].

num_packed_regions specifies the number of packed regions when constituent_picture_matching_flag is equal to 0. The value of num_packed_regions shall be greater than 0. When constituent_picture_matching_flag is equal to 1, the total number of packed regions is equal to num_packed_regions * 2, and the information in each entry of the loop of num_packed_regions entries applies to each constituent picture of the projected picture and the packed picture.

proj_picture_width and **proj_picture_height** specify the width and height, respectively, of the projected picture, in relative projected picture sample units.

The values of proj_picture_width and proj_picture_height shall both be greater than 0.

packed_picture_width and **packed_picture_height** specify the width and height, respectively, of the packed picture, in relative packed picture sample units.

The values of packed_picture_width and packed_picture_height shall both be greater than 0.

It is a requirement of bitstream conformance that packed_picture_width and packed_picture_height shall have such values that packed_picture_width is an integer multiple of cropPicWidth and packed_picture_height is an integer multiple of cropPicHeight, where cropPicWidth and cropPicHeight are the width and height, respectively, of the cropped decoded picture.

rwp_reserved_zero_4bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for rwp_reserved_zero_4bits[i] are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of rwp_reserved_zero_4bits[i].

rwp_transform_type[i] specifies the rotation and mirroring to be applied to the i-th packed region to remap to the i-th projected region. When rwp_transform_type[i] specifies both rotation and mirroring, rotation applies before mirroring. The values of rwp_transform_type[i] are specified in Table D.22.

Table D.22 – rwp_transform_type[i] values

Value	Description
0	no transform
1	mirroring horizontally
2	rotation by 180 degrees (anticlockwise)
3	rotation by 180 degrees (anticlockwise) before mirroring horizontally
4	rotation by 90 degrees (anticlockwise) before mirroring horizontally
5	rotation by 90 degrees (anticlockwise)
6	rotation by 270 degrees (anticlockwise) before mirroring horizontally
7	rotation by 270 degrees (anticlockwise)

rwp_guard_band_flag[i] equal to 0 specifies that the i-th packed region does not have a guard band. rwp_guard_band_flag[i] equal to 1 specifies that the i-th packed region has a guard band.

proj_region_width[i], **proj_region_height**[i], **proj_region_top**[i] and **proj_region_left**[i] specify the width, height, top sample row, and the left-most sample column, respectively, of the i-th projected region, either within the projected picture (when constituent_picture_matching_flag is equal to 0) or within the constituent picture of the projected picture (when constituent_picture_matching_flag is equal to 1).

proj_region_width[i], proj_region_height[i], proj_region_top[i], and proj_region_left[i] are indicated in relative projected picture sample units.

NOTE 1 – Two projected regions may partially or entirely overlap with each other.

packed_region_width[i], **packed_region_height**[i], **packed_region_top**[i], and **packed_region_left**[i] specify the width, height, the top luma sample row, and the left-most luma sample column, respectively, of the packed region, either within the region-wise packed picture (when constituent_picture_matching_flag is equal to 0) or within each constituent picture of the region-wise packed picture (when constituent_picture_matching_flag is equal to 1).

packed_region_width[i], packed_region_height[i], packed_region_top[i], and packed_region_left[i] are indicated in relative region-wise packed picture sample units. packed_region_width[i], packed_region_height[i], packed_region_top[i], and packed_region_left[i] shall represent integer horizontal and vertical coordinates of luma sample units within the cropped decoded pictures.

NOTE 2 – Two packed regions may partially or entirely overlap with each other.

rwp_left_guard_band_width[i] specifies the width of the guard band on the left side of the i-th packed region in relative region-wise packed picture sample units. When chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), rwp_left_guard_band_width[i] shall correspond to an even number of luma samples within the cropped decoded picture.

rwp_right_guard_band_width[i] specifies the width of the guard band on the right side of the i-th packed region in relative region-wise packed picture sample units. When chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), rwp_right_guard_band_width[i] shall correspond to an even number of luma samples within the cropped decoded picture.

rwp_top_guard_band_height[i] specifies the height of the guard band above the i-th packed region in relative region-wise packed picture sample units. When chroma_format_idc is equal to 1 (4:2:0 chroma format), rwp_top_guard_band_height[i] shall correspond to an even number of luma samples within the cropped decoded picture.

rwp_bottom_guard_band_height[i] specifies the height of the guard band below the i-th packed region in relative region-wise packed picture sample units. When chroma_format_idc is equal to 1 (4:2:0 chroma format), rwp_bottom_guard_band_height[i] shall correspond to an even number of luma samples within the cropped decoded picture.

When rwp_guard_band_flag[i] is equal to 1, rwp_left_guard_band_width[i], rwp_right_guard_band_width[i], rwp_top_guard_band_height[i], or rwp_bottom_guard_band_height[i] shall be greater than 0.

The i-th packed region as specified by this SEI message shall not overlap with any other packed region specified by the same SEI message or any guard band specified by the same SEI message.

The guard bands associated with the i-th packed region, if any, as specified by this SEI message shall not overlap with any packed region specified by the same SEI message or any other guard bands specified by the same SEI message.

rwp_guard_band_not_used_for_pred_flag[i] equal to 0 specifies that the guard bands may or may not be used in the inter prediction process. rwp_guard_band_not_used_for_pred_flag[i] equal to 1 specifies that the sample values of the guard bands are not used in the inter prediction process.

NOTE 3 – When rwp_guard_band_not_used_for_pred_flag[i] is equal to 1, the sample values within guard bands in cropped decoded pictures can be rewritten even if the cropped decoded pictures were used as references for inter prediction of subsequent pictures to be decoded. For example, the content of a packed region can be seamlessly expanded to its guard band with decoded and re-projected samples of another packed region.

rwp_guard_band_type[i][j] indicates the type of the guard bands for the i-th packed region as follows, with j equal to 0, 1, 2, or 3 indicating that the semantics below apply to the left, right, top, or bottom edge, respectively, of the packed region:

– rwp_guard_band_type[i][j] equal to 0 indicates that the content of the guard bands in relation to the content of the packed regions is unspecified. When rwp_guard_band_not_used_for_pred_flag[i] is equal to 0, rwp_guard_band_type[i][j] shall not be equal to 0.

– rwp_guard_band_type[i][j] equal to 1 indicates that the content of the guard bands suffices for interpolation of sample values at sub-pel sample fractional locations within the packed region and less than one sample outside of the boundary of the packed region.

  NOTE 4 – rwp_guard_band_type[i][j] equal to 1 can be used when the boundary samples of a packed region have been copied horizontally or vertically to the guard band.

– rwp_guard_band_type[i][j] equal to 2 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at quality that gradually changes from the picture quality of the packed region to that of the spherically adjacent packed region.

– rwp_guard_band_type[i][j] equal to 3 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at a similar picture quality as within the packed region.

– rwp_guard_band_type[i][j] values greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value of rwp_guard_band_type[i][j] when the value is greater than 3 as equivalent to the value 0.

rwp_guard_band_reserved_zero_3bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for rwp_guard_band_reserved_zero_3bits[i] are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of rwp_guard_band_reserved_zero_3bits[i].

The variables NumPackedRegions, PackedRegionLeft[n], PackedRegionTop[n], PackedRegionWidth[n], PackedRegionHeight[n], ProjRegionLeft[n], ProjRegionTop[n], ProjRegionWidth[n], ProjRegionHeight[n], and TransformType[n] are derived as follows:

– For n in the range of 0 to num_packed_regions − 1, inclusive, the following applies:
  – PackedRegionLeft[n] is set equal to packed_region_left[n].
  – PackedRegionTop[n] is set equal to packed_region_top[n].
  – PackedRegionWidth[n] is set equal to packed_region_width[n].
  – PackedRegionHeight[n] is set equal to packed_region_height[n].
  – ProjRegionLeft[n] is set equal to proj_region_left[n].
  – ProjRegionTop[n] is set equal to proj_region_top[n].
  – ProjRegionWidth[n] is set equal to proj_region_width[n].
  – ProjRegionHeight[n] is set equal to proj_region_height[n].
  – TransformType[n] is set equal to rwp_transform_type[n].
– If constituent_picture_matching_flag is equal to 0, the following applies:
  – NumPackedRegions is set equal to num_packed_regions.
– Otherwise (constituent_picture_matching_flag is equal to 1), the following applies:
  – NumPackedRegions is set equal to 2 * num_packed_regions.

- When TopBottomFlag is equal to 1, the following applies:
  - projLeftOffset and packedLeftOffset are both set equal to 0.
  - projTopOffset is set equal to proj_picture_height / 2 and packedTopOffset is set equal to packed_picture_height / 2.
- When SideBySideFlag is equal to 1, the following applies:
  - projLeftOffset is set equal to proj_picture_width / 2 and packedLeftOffset is set equal to packed_picture_width / 2.
  - projTopOffset and packedTopOffset are both set equal to 0.
- For n in the range of NumPackedRegions / 2 to NumPackedRegions − 1, inclusive, the following applies:
  - nIdx is set equal to n − NumPackedRegions / 2.
  - PackedRegionLeft[n] is set equal to packed_region_left[nIdx] + packedLeftOffset.
  - PackedRegionTop[n] is set equal to packed_region_top[nIdx] + packedTopOffset.
  - PackedRegionWidth[n] is set equal to packed_region_width[nIdx].
  - PackedRegionHeight[n] is set equal to packed_region_height[nIdx].
  - ProjRegionLeft[n] is set equal to proj_region_left[nIdx] + projLeftOffset.
  - ProjRegionTop[n] is set equal to proj_region_top[nIdx] + projTopOffset.
  - ProjRegionWidth[n] is set equal to proj_region_width[nIdx].
  - ProjRegionHeight[n] is set equal to proj_region_height[nIdx].
  - TransformType[n] is set equal to rwp_transform_type[nIdx].

For each value of n in the range of 0 to NumPackedRegions − 1, inclusive, the values of ProjRegionWidth[n], ProjRegionHeight[n], ProjRegionTop[n], and ProjRegionLeft[n] are constrained as follows:

- ProjRegionWidth[n] shall be in the range of 1 to proj_picture_width, inclusive.
- ProjRegionHeight[n] shall be in the range of 1 to proj_picture_height, inclusive.
- ProjRegionLeft[n] shall be in the range of 0 to proj_picture_width − 1, inclusive.
- ProjRegionTop[n] shall be in the range of 0 to proj_picture_height − 1, inclusive.
- If ProjRegionTop[n] is less than proj_picture_height / VerDiv1, the sum of ProjRegionTop[n] and ProjRegionHeight[n] shall be less than or equal to proj_picture_height / VerDiv1. Otherwise, the sum of ProjRegionTop[n] and ProjRegionHeight[n] shall be less than or equal to proj_picture_height / VerDiv1 * 2.

For each value of n in the range of 0 to NumPackedRegions − 1, inclusive, the values of PackedRegionWidth[n], PackedRegionHeight[n], PackedRegionTop[n], and PackedRegionLeft[n] are constrained as follows:

- PackedRegionWidth[n] shall be in the range of 1 to packed_picture_width, inclusive.
- ProjRegionHeight[n] shall be in the range of 1 to packed_picture_height, inclusive.
- PackedRegionLeft[n] shall be in the range of 0 to packed_picture_width − 1, inclusive.
- PackedRegionTop[n] shall be in the range of 0 to packed_picture_height − 1, inclusive.
- If PackedRegionLeft[n] is less than packed_picture_width / HorDiv1, the sum of PackedRegionLeft[n] and PackedRegionWidth[n] shall be less than or equal to packed_picture_width / HorDiv1. Otherwise, the sum of PackedRegionLeft[n] and PackedRegionWidth[n] shall be less than or equal to packed_picture_width / HorDiv1 * 2.
- If PackedRegionTop[n] is less than packed_picture_height / VerDiv1, the sum of PackedRegionTop[n] and PackedRegionHeight[n] shall be less than or equal to packed_picture_height / VerDiv1. Otherwise, the sum of PackedRegionTop[n] and PackedRegionHeight[n] shall be less than or equal to packed_picture_height / VerDiv1 * 2.
- When chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), PackedRegionLeft[n] shall correspond to an even horizontal coordinate value of luma sample units, and PackedRegionWidth[n] shall correspond to an even number of luma samples, both within the decoded picture.
- When chroma_format_idc is equal to 1 (4:2:0 chroma format), PackedRegionTop[n] shall correspond to an even vertical coordinate value of luma sample units, and ProjRegionHeight[n] shall correspond to an even number of luma samples, both within the decoded picture.

D.3.41.6 Omnidirectional viewport SEI message semantics

The omnidirectional viewport SEI message specifies the coordinates of one or more regions of sphere-coordinate geometry, bounded by four great circles, corresponding to viewports recommended for display when the user does not have control of the viewing orientation or has released control of the viewing orientation.

When an effectively applicable frame packing arrangement SEI message, as specified in clause D.3.41.1 or D.3.41.2, that applies to the picture is present, the information indicated by the omnidirectional viewport SEI message applies to both views.

omni_viewport_id contains an identifying number that may be used to identify the purpose of the one or more recommended viewport regions.

omni_viewport_id equal to 0 indicates that the recommended viewports are per "director’s cut", i.e., a viewport suggested according to the creative intent of the content author or content provider. omni_viewport_id equal to 1 indicates that the recommended viewports are selected based on measurements of viewing statistics.

Values of omni_viewport_id from 2 to 511, inclusive, may be used as determined by the application. Values of omni_viewport_id from 512 to 1 023 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of omni_viewport_id in the range of 512 to 1 023, inclusive, shall ignore it.

omni_viewport_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous omnidirectional viewport SEI message in output order. omni_viewport_cancel_flag equal to 0 indicates that omnidirectional viewport information follows.

omni_viewport_persistence_flag specifies the persistence of the omnidirectional viewport SEI message for the current layer.

omni_viewport_persistence_flag equal to 0 specifies that the omnidirectional viewport SEI message applies to the current decoded picture only.

Let picA be the current picture. omni_viewport_persistence_flag equal to 1 specifies that the omnidirectional viewport SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.
– The bitstream ends.
– A picture picB in the current layer in an access unit containing an omnidirectional viewport SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

When an equirectangular projection SEI message with erp_cancel_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 is not present in the CLVS that applies to the current picture and precedes the omnidirectional viewport SEI message in decoding order, an omnidirectional viewport SEI message with omni_viewport_cancel_flag equal to 0 shall not be present in the CLVS that applies to the current picture. Decoders shall ignore omnidirectional viewport SEI messages with omni_viewport_cancel_flag equal to 0 that do not follow, in decoding order, an equirectangular projection SEI message with erp_cancel_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 in the CLVS that applies to the current picture.

omni_viewport_cnt_minus1 plus 1 specifies the number of recommended viewport regions that are indicated by the SEI message.

When omni_viewport_cnt_minus1 is greater than 0 and there is no information provided by external means not specified in this Specification on which recommended viewport is suggested to be displayed, the following applies:

– When omni_viewport_id is equal to 0 or 1, the 0-th recommended viewport is suggested to be displayed when the user does not have control of the viewing orientation or has released control of the viewing orientation.
– When omni_viewport_id is equal to 0, between any two recommended viewports per director's cut, the i-th recommended viewport has higher priority than the j-th recommended viewport for any values of i and j when i is less than j. The 0-th recommended viewport per director's cut has the highest priority.
– When omni_viewport_id is equal to 1, between any two recommended viewports, the i-th recommended viewport has higher popularity, among some selection of candidate viewports, than the j-th recommended viewport for any values of i and j when i is less than j. The 0-th most-viewed recommended viewport has the highest popularity. The selection of the candidate viewports is outside the scope of this Specification.

omni_viewport_azimuth_centre[i] and **omni_viewport_elevation_centre**[i] indicate the centre of the i-th recommended viewport region, in units of 2^{-16} degrees relative to the global coordinate axes. The value of

omni_viewport_azimuth_centre[i] shall be in the range of $-180 * 2^{16}$ (i.e., −11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. The value of omni_viewport_elevation_centre[i] shall be in the range of $-90 * 2^{16}$ (i.e., −5 898 240) to $90 * 2^{16}$ (i.e., 5 898 240), inclusive.

omni_viewport_tilt_centre[i] indicates the tilt angle of the i-th recommended viewport region, in units of 2^{-16} degrees. The value of omni_viewport_tilt_centre[i] shall be in the range of $-180 * 2^{16}$ (i.e., −11 796 480) to $2^{16} - 1$ (i.e., 11 796 479), inclusive.

omni_viewport_hor_range[i] indicates the azimuth range of the i-th recommended viewport region, in units of 2^{-16} degrees. The value of omni_viewport_hor_range[i] shall be in the range of 1 to $360 * 2^{16}$ (i.e., 23 592 960), inclusive.

omni_viewport_ver_range[i] indicates the elevation range of the i-th recommended viewport region, in units of 2^{-16} degrees. The value of omni_viewport_ver_range[i] shall be in the range of 1 to $180 * 2^{16}$ (i.e., 11 796 480), inclusive.

D.3.41.7 Sample location remapping process

D.3.41.7.1 General

To remap colour sample locations of a region-wise packed picture to a unit sphere, the following ordered steps are applied:

1. A region-wise packed picture is obtained as the cropped decoded picture by decoding a coded picture. For purposes of interpretation of chroma samples, the input to the indicated remapping process is the set of decoded sample values after applying an (unspecified) upsampling conversion process to the 4:4:4 colour sampling format as necessary when chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format). This (unspecified) upsampling process should account for the relative positioning relationship between the luma and chroma samples as indicated by chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field, when present.
2. If region-wise packing is indicated, the sample locations of the region-wise packed picture are converted to sample locations of the respective projected picture as specified in clause D.3.41.7.4. Otherwise, the projected picture is identical to the region-wise packed picture.
3. If frame packing is indicated, the sample locations of the projected picture are converted to sample locations of the respective constituent picture of the projected picture, as specified in clause D.3.41.7.5. Otherwise, the constituent picture of the projected picture is identical to the projected picture.
4. The sample locations of a constituent picture of the projected picture are converted to sphere coordinates relative to the local coordinate axes, as specified in clause D.3.41.7.2.
5. If rotation is indicated, the sphere coordinates relative to the local coordinate axes are converted to sphere coordinates relative to the global coordinate axes, as specified in clause D.3.41.7.3. Otherwise, the global coordinate axes are identical to the local coordinate axes.

The overall process for mapping of luma sample locations within a region-wise packed picture to sphere coordinates relative to the global coordinate axes is normatively specified in clause D.3.41.7.5.

For each region-wise packed picture corresponding to a decoded picture, the following applies:

– When an equirectangular projection SEI message with erp_cancel_flag equal to 0 that applies to the picture is present, ErpFlag is set equal to 1, and CmpFlag is set equal to 0.

– When a cubemap projection SEI message with cmp_cancel_flag equal to 0 that applies to the picture is present, CmpFlag is set equal to 1, and ErpFlag is set equal to 0.

– If a sphere rotation SEI message with sphere_rotation_cancel_flag equal to 0 that applies to the picture is present, RotationFlag is set equal to 1, and RotationYaw, RotationPitch, and RotationRoll are set equal to yaw_rotation $\div 2^{16}$, pitch_rotation $\div 2^{16}$, and roll_rotation $\div 2^{16}$, respectively.

– Otherwise, RotationFlag is set equal to 0.

– If a frame packing arrangement SEI message with frame_packing_arrangement_cancel_flag equal to 0 that applies to the picture is not present, StereoFlag, TopBottomFlag, and SideBySideFlag are all set equal to 0, HorDiv1 is set equal to 1, and VerDiv1 is set equal to 1.

– Otherwise, the following applies:

  – StereoFlag is set equal to 1.

  – If the value of frame_packing_arrangement_type of the frame packing arrangement SEI message is equal to 3, TopBottomFlag is set equal to 0, SideBySideFlag is set equal to 1, HorDiv1 is set equal to 2 and VerDiv1 is set equal to 1.

    – Otherwise, if the value of frame_packing_arrangement_type of the frame packing arrangement SEI message is equal to 4, TopBottomFlag is set equal to 1, SideBySideFlag is set equal to 0, HorDiv1 is set equal to 1, and VerDiv1 is set equal to 2.
    – Otherwise, TopBottomFlag is set equal to 0, SideBySideFlag is set equal to 0, HorDiv1 is set equal to 1, and VerDiv1 is set equal to 1.
– If a region-wise packing SEI message with rwp_cancel_flag equal to 0 that applies to the picture is not present, RegionWisePackingFlag is set equal to 0, and ConstituentPicWidth and ConstituentPicHeight are set to be equal to cropPicWidth / HorDiv1 and cropPicHeight / VerDiv1, respectively, where cropPicWidth and cropPicHeight are the width and height, respectively, of the cropped decoded picture.
– Otherwise, RegionWisePackingFlag is set equal to 1, and ConstituentPicWidth and ConstituentPicHeight are set equal to proj_picture_width / HorDiv1 and proj_picture_height / VerDiv1, respectively.

To remap colour sample locations of a fisheye video picture to a unit sphere, the sample locations in each of the active regions is converted to locations on the unit sphere as specified in clause D.3.41.7.7.

D.3.41.7.2 Projection for one sample location

Inputs to this process are:

– pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units, and
– the centre point of a sample location (hPos, vPos) along the horizontal and vertical axes, respectively, in relative projected picture sample units, where hPos and vPos may have non-integer real values.

Outputs of this process are:

– sphere coordinates (ϕ, θ) for the sample location in degrees relative to the local coordinate axes.

The projection for a sample location is derived as follows:

– If ErpFlag is equal to 1, the following applies:
    – If RegionWisePackingFlag is equal to 0 and erp_guard_band_flag is equal to 1, the following applies:

$$\text{hPos}' = \text{hPos} - \text{erp\_left\_guard\_band\_width} \quad \text{(D-54)}$$
$$\text{pictureWidth} = \text{pictureWidth} - \text{erp\_left\_guard\_band\_width} - \text{erp\_right\_guard\_band\_width}$$

    – Otherwise, the following applies:

$$\text{hPos}' = \text{hPos} \quad \text{(D-55)}$$

    – The following applies:

$$\phi = 180 - \text{hPos}' * (360 \div \text{pictureWidth}) \quad \text{(D-56)}$$
$$\theta = 90 - \text{vPos} * (180 \div \text{pictureHeight})$$

– Otherwise (CmpFlag is equal to 1), it is a requirement of bitstream conformance that pictureWidth shall be a multiple of 3 and pictureHeight shall be a multiple of 2, and that pictureWidth / 3 shall be equal to pictureHeight / 2, and the following applies:

```
lw = pictureWidth / 3
lh = pictureHeight / 2
w = Floor( hPos ÷ lw )
h = Floor( vPos ÷ lh )
tmpHorVal = hPos − w * lw
tmpVerVal = vPos − h * lh
hPos′ = −( 2 * tmpHorVal ÷ lw ) + 1
vPos′ = −( 2 * tmpVerVal ÷ lh ) + 1
if( w = = 1 && h = = 0 ) { /* positive x front face */
    x = 1.0
    y = hPos′
    z = vPos′
} else if( w = = 1 && h = = 1 ) { /* negative x back face */
    x = −1.0
```

$$
\begin{aligned}
&\quad y = -vPos' \\
&\quad z = -hPos' \\
&\} \text{ else if(} w == 2 \text{ \&\& } h == 1 \text{) \{ /* positive z top face */} \\
&\quad x = -hPos' \\
&\quad y = -vPos' \\
&\quad z = 1.0 \\
&\} \text{ else if(} w == 0 \text{ \&\& } h == 1 \text{) \{ /* negative z bottom face */} \\
&\quad x = hPos' \\
&\quad y = -vPos' \\
&\quad z = -1.0 \\
&\} \text{ else if(} w == 0 \text{ \&\& } h == 0 \text{) \{ /* positive y left face */} \\
&\quad x = -hPos' \\
&\quad y = 1.0 \\
&\quad z = vPos' \\
&\} \text{ else \{ /* (} w == 2 \text{ \&\& } h == 0 \text{), /* negative y right face */} \\
&\quad x = hPos' \\
&\quad y = -1.0 \\
&\quad z = vPos' \\
&\} \\
&\phi = \text{Atan2}(\, y, x \,) * 180 \div \pi \\
&\theta = \text{Asin}(\, z \div \text{Sqrt}(\, x^2 + y^2 + z^2 \,)\,) * 180 \div \pi
\end{aligned}
\tag{D-57}
$$

D.3.41.7.3 Conversion from the local coordinate axes to the global coordinate axes

Inputs to this process are:

– rotation_yaw (α_d), rotation_pitch (β_d), rotation_roll (γ_d), all in units of degrees, and

– sphere coordinates (ϕ_d, θ_d) relative to the local coordinate axes.

Outputs of this process are:

– sphere coordinates (ϕ', θ') relative to the global coordinate axes.

The outputs are derived as follows:

$$
\begin{aligned}
&\phi = \phi_d * \pi \div 180 \\
&\theta = \theta_d * \pi \div 180 \\
&\alpha = \alpha_d * \pi \div 180 \\
&\beta = \beta_d * \pi \div 180 \\
&\gamma = \gamma_d * \pi \div 180 \\
&x_1 = \text{Cos}(\,\phi\,) * \text{Cos}(\,\theta\,) \\
&y_1 = \text{Sin}(\,\phi\,) * \text{Cos}(\,\theta\,) \\
&z_1 = \text{Sin}(\,\theta\,) \\
&x_2 = \text{Cos}(\,\beta\,) * \text{Cos}\,(\,\alpha\,) * x_1 - \text{Cos}(\,\beta\,) * \text{Sin}(\,\alpha\,) * y_1 + \text{Sin}(\,\beta\,) * z_1 \\
&y_2 = (\,\text{Cos}(\,\gamma\,) * \text{Sin}(\,\alpha\,) + \text{Sin}(\,\gamma\,) * \text{Sin}(\,\beta\,) * \text{Cos}(\,\alpha\,)\,) * x_1 + \\
&\quad (\,\text{Cos}(\,\gamma\,) * \text{Cos}(\,\alpha\,) - \text{Sin}(\,\gamma\,) * \text{Sin}(\,\beta\,) * \text{Sin}(\,\alpha\,)\,) * y_1 - \\
&\quad \text{Sin}(\,\gamma\,) * \text{Cos}(\,\beta\,) * z_1 \\
&z_2 = (\,\text{Sin}(\,\gamma\,) * \text{Sin}(\,\alpha\,) - \text{Cos}(\,\gamma\,) * \text{Sin}(\,\beta\,) * \text{Cos}(\,\alpha\,)\,) * x_1 + \\
&\quad (\,\text{Sin}(\,\gamma\,) * \text{Cos}(\,\alpha\,) + \text{Cos}(\,\gamma\,) * \text{Sin}(\,\beta\,) * \text{Sin}(\,\alpha\,)\,) * y_1 + \\
&\quad \text{Cos}(\,\gamma\,) * \text{Cos}(\,\beta\,) * z_1 \\
&\phi' = \text{Atan2}(\,y_2, x_2\,) * 180 \div \pi \\
&\theta' = \text{Asin}(\,z_2\,) * 180 \div \pi
\end{aligned}
\tag{D-58}
$$

D.3.41.7.4 Conversion of sample locations for rectangular region-wise packing

Inputs to this process are:

– sample location (x, y) within the packed region, where x and y are in relative packed picture sample units, while the sample location is at an integer sample location within the packed picture,

– the width and the height (projRegWidth, projRegHeight) of the projected region, in relative projected picture sample units,

– the width and the height (packedRegWidth, packedRegHeight) of the packed region, in relative packed picture sample units,

– transform type (transformType), and

– offset values for the sampling position (offsetX, offsetY) in the range of 0, inclusive, to 1, exclusive, in horizontal and vertical relative packed picture sample units, respectively.

NOTE – offsetX and offsetY both equal to 0.5 indicate a sampling position that is in the centre point of a sample in packed picture sample units.

Outputs of this process are:

– the centre point of the sample location (hPos, vPos) within the projected region in relative projected picture sample units, where hPos and vPos may have non-integer real values.

The outputs are derived as follows:

```
if( transformType  = =  0 || transformType  = =  1 || transformType  = =  2 || transformType  = =
  3 ) {
     horRatio = projRegWidth ÷ packedRegWidth
     verRatio = projRegHeight ÷ packedRegHeight
} else if( transformType  = =  4 || transformType  = =  5 || transformType  = =  6 ||
     transformType  = =  7 ) {
     horRatio = projRegWidth ÷ packedRegHeight
     verRatio = projRegHeight ÷ packedRegWidth
}
if( transformType  = =  0 ) {
     hPos = horRatio * ( x + offsetX )
     vPos = verRatio * ( y + offsetY )
} else if( transformType  = =  1 ) {
     hPos = horRatio * ( packedRegWidth − x − offsetX )
     vPos = verRatio * ( y + offsetY )
} else if( transformType  = =  2 ) {
     hPos = horRatio * ( packedRegWidth − x − offsetX )
     vPos = verRatio * ( packedRegHeight − y − offsetY )                                    (D-59)
} else if( transformType  = =  3 ) {
     hPos = horRatio * ( x + offsetX )
     vPos = verRatio * ( packedRegHeight − y − offsetY )
} else if( transformType  = =  4 ) {
     hPos = horRatio * ( y + offsetY )
     vPos = verRatio * ( x + offsetX )
} else if( transformType  = =  5 ) {
     hPos = horRatio * ( y + offsetY )
     vPos = verRatio * ( packedRegWidth − x − offsetX )
} else if( transformType  = = 6 ) {
     hPos = horRatio * ( packedRegHeight − y − offsetY )
     vPos = verRatio * ( packedRegWidth − x − offsetX )
} else if( transformType  = =  7 ) {
     hPos = horRatio * ( packedRegHeight − y − offsetY )
     vPos = verRatio * ( x + offsetX )
}
```

D.3.41.7.5 Mapping of luma sample locations within a cropped decoded picture to sphere coordinates relative to the global coordinate axes

This clause specifies the mapping of luma sample locations within a cropped decoded picture to sphere coordinates relative to the global coordinate axes.

offsetX is set equal to 0.5 and offsetY is set equal to 0.5.

If RegionWisePackingFlag is equal to 1, the following applies for each packed region n in the range of 0 to NumPackedRegions − 1, inclusive:

- For each sample location (xPackedPicture, yPackedPicture) belonging to the n-th packed region, the following applies:
  - The corresponding sample location (xProjPicture, yProjPicture) of the projected picture is derived as follows:
    - x is set equal to xPackedPicture − PackedRegionLeft[n].
    - y is set equal to yPackedPicture − PackedRegionTop[n].
    - Clause D.3.41.7.4 is invoked with x, y, PackedRegionWidth[n], PackedRegionHeight[n], ProjRegion Width[n], ProjRegionHeight[n], TransformType[n], offsetX and offsetY as inputs, and the output is assigned to sample location (hPos, vPos).
    - xProjPicture is set equal to ProjRegionLeft[n] + hPos.
    - When StereoFlag is equal to 0 or TopBottomFlag is equal to 1, and when xProjPicture is greater than or equal to proj_picture_width, xProjPicture is set equal to xProjPicture − proj_picture_width.
    - When SideBySideFlag is equal to 1, the following applies:
      - When ProjRegionLeft[n] is less than proj_picture_width / 2 and xProjPicture is greater than or equal to proj_picture_width / 2, xProjPicture is set equal to xProjPicture − proj_picture_width / 2.
      - When ProjRegionLeft[n] is greater than or equal to proj_picture_width / 2 and xProjPicture is greater than or equal to proj_picture_width, xProjPicture is set equal to xProjPicture − proj_picture_width / 2.
    - yProjPicture is set equal to ProjRegionTop[n] + vPos.
  - Clause D.3.41.7.6 is invoked with xProjPicture, yProjPicture, ConstituentPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent picture index (for frame-packed stereoscopic video) for the luma sample location (xPackedPicture, yPackedPicture) belonging to the n-th packed region in the decoded picture.

Otherwise (RegionWisePackingFlag is equal 0), the following applies for each sample location (x, y) that is not an equirectangular projection guard band sample within the cropped decoded picture, where a sample location (x, y) is an equirectangular projection guard band sample when and only when ErpFlag is equal to 1, x is in the range of 0 to erp_left_guard_band_width − 1, inclusive, or ConstituentPicWidth − erp_right_guard_band_width to ConstituentPicWidth − 1, inclusive, and y is in the range of 0 to ConstituentPicHeight − 1, inclusive:

- xProjPicture is set equal to x + offsetX.
- yProjPicture is set equal to y + offsetY.
- If ErpFlag is equal to 0, projPicWidth is set equal to ConstituentPicWidth. Otherwise (ErpFlag is equal to 1), projPicWidth is set equal to ConstituentPicWidth − (erp_left_guard_band_width + erp_right_guard_band_width).
- Clause D.3.41.7.6 is invoked with xProjPicture, yProjPicture, projPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent picture index (for frame-packed stereoscopic video) for the sample location (x, y) within the region-wise packed picture.

D.3.41.7.6 Conversion from a sample location in a projected picture to sphere coordinates relative to the global coordinate axes

Inputs to this process are:

- the centre point of a sample location (xProjPicture, yProjPicture) within a projected picture, where xProjPicture and yProjPicture are in relative projected picture sample units and may have non-integer real values, and
- pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units.

Outputs of this process are:

- sphere coordinates (azimuthGlobal, elevationGlobal), in units of degrees relative to the global coordinate axes, and
- when StereoFlag is equal to 1, the index of the constituent picture (constituentPicture) equal to 0 or 1.

The outputs are derived with the following ordered steps:

1. constituentPicture, xProjPicture, and yProjPicture are conditionally set as follows:
   - If xProjPicture is greater than or equal to pictureWidth or yProjPicture is greater than or equal to pictureHeight, the following applies:

- constituentPicture is set equal to 1.
- When xProjPicture is greater than or equal to pictureWidth, xProjPicture is set to xProjPicture − pictureWidth.
- When yProjPicture is greater than or equal to pictureHeight, yProjPicture is set to yProjPicture − pictureHeight.

– Otherwise, constituentPicture is set equal to 0.

2. Clause D.3.41.7.2 is invoked with pictureWidth, pictureHeight, xProjPicture, and yProjPicture as inputs, and the output is assigned to azimuthLocal, elevationLocal.
3. azimuthGlobal and elevationGlobal are set as follows:
   - If RotationFlag is equal to 1, clause D.3.41.7.3 is invoked with azimuthLocal, elevationLocal, RotationYaw, RotationPitch, and RotationRoll as inputs, and the output is assigned to azimuthGlobal and elevationGlobal.
   - Otherwise, azimuthGlobal is set equal to azimuthLocal and elevationGlobal is set equal to elevationLocal.

D.3.41.7.7 Conversion from a sample location of an active area to sphere coordinates relative to the global coordinate axes

Inputs to this process are:

- the sample location (x, y) in units of luma samples,
- the centre location (x_c, y_c) and the radius (r_c) of the circular region that contains the i-th active area, given by fisheye_circular_region_centre_x[i], fisheye_circular_region_centre_y[i], and fisheye_circular_region_radius[i], respectively, all in units of 2^{-16} luma samples,
- the field of view (θ_v) of the lens corresponding to the i-th active area, given by fisheye_field_of_view[i], in units of 2^{-16} degrees,
- the rotation parameters (α_c, β_c, γ_c), given by fisheye_camera_centre_azimuth[i], fisheye_camera_centre_elevation[i], and fisheye_camera_centre_tilt[i], respectively, all in units of 2^{-16} degrees, and
- the number of polynomial coefficients numCoeffs and the polynomial coefficients coeffVal[j] (for j ranging from 0 to numCoeffs − 1, inclusive) of the i-th active area, given by fisheye_num_polynomial_coeffs[i] and fisheye_polynomial_coeff[i][j] (for j ranging from 0 to fisheye_num_polynomial_coeffs[i] − 1, inclusive), respectively.

Outputs of this process are:

- sphere coordinates (ϕ, θ) relative to the global coordinate axes.

The method of converting a sample location of an active area to sphere coordinates is determined as follows:

- If numCoeffs is equal to 0, there is only one method of converting a sample location of an active area to sphere coordinates that is specified, which is to not use polynomial coefficients.
- Otherwise (numCoeffs is not equal to 0), there are two methods of converting a sample location of an active area to sphere coordinates that are specified, which are to not use polynomial coefficients or to use polynomial coefficients. The method using polynomial coefficients is preferred, as this method is intended to provide a more precise model of the fisheye characteristics. However, the other method may also be appropriate for some uses, as it provides a single conversion process that can be used regardless of whether numCoeffs is equal to 0 or not. This Specification does not prescribe which of the two methods is to be used in this case.

The outputs are derived as follows:

- If polynomial coefficients are not used, the angle ϕ' is derived by

$$\phi' = (\ \mathrm{Sqrt}(\ (\ x - x_c \div 2^{16}\ )^2 + (\ y - y_c \div 2^{16}\ )^2\ ) \div (\ r_c \div 2^{16}\ )\ ) * (\ \theta_v \div 2^{16} * \pi \div 180\ ) \div 2 \qquad \text{(D-60)}$$

- Otherwise (polynomial coefficients are used), the angle ϕ' is derived by

$$\phi' = \sum_{j=0}^{numCoeffs-1} (\ (\ \mathrm{coeffVal}[\ j\ ] * 2^{-24}\ ) * (\ \mathrm{Sqrt}(\ (\ x - x_c * 2^{-16}\ )^2 + (\ y - y_c * 2^{-16}\ )^2\ ) \div (\ r_c * 2^{-16}\ )\ )^j\ ) \qquad \text{(D-61)}$$

The outputs are then derived as follows:

$$\theta' = \mathrm{Atan2}(\ y - y_c \div 2^{16},\ x - x_c \div 2^{16}\ )$$
$$x_1 = \mathrm{Cos}(\ \phi'\ )$$
$$y_1 = \mathrm{Sin}(\ \phi'\ ) * \mathrm{Cos}(\ \theta'\ )$$
$$z_1 = \mathrm{Sin}(\ \phi'\ ) * \mathrm{Sin}(\ \theta'\ )$$
$$\alpha = (\ \alpha_c \div 2^{16}\ ) * \pi \div 180$$
$$\beta = (\ \beta_c \div 2^{16}\ ) * \pi \div 180$$
$$\gamma = (\ \gamma_c \div 2^{16}\ ) * \pi \div 180$$
$$x_2 = \mathrm{Cos}(\ \beta\ ) * \mathrm{Cos}\ (\ \gamma\ ) * x_1 - \mathrm{Cos}(\ \beta\ ) * \mathrm{Sin}(\ \gamma\ ) * y_1 + \mathrm{Sin}(\ \beta\ ) * z_1 \qquad \text{(D-62)}$$
$$y_2 = (\ \mathrm{Cos}(\ \alpha\ ) * \mathrm{Sin}(\ \gamma\ ) + \mathrm{Sin}(\ \alpha\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Cos}(\ \gamma\ )\ ) * x_1 + (\ \mathrm{Cos}(\ \alpha\ ) * \mathrm{Cos}(\ \gamma\ ) - \mathrm{Sin}(\ \alpha\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Sin}(\ \gamma\ )\ ) * y_1 - \mathrm{Sin}(\ \alpha\ ) * \mathrm{Cos}(\ \beta\ ) * z_1$$
$$z_2 = (\ \mathrm{Sin}(\ \alpha\ ) * \mathrm{Sin}(\ \gamma\ ) - \mathrm{Cos}(\ \alpha\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Cos}(\ \gamma\ )\ ) * x_1 + (\ \mathrm{Sin}(\ \alpha\ ) * \mathrm{Cos}(\ \gamma\ ) + \mathrm{Cos}(\ \alpha\ ) * \mathrm{Sin}(\ \beta\ ) * \mathrm{Sin}(\ \gamma\ )\ ) * y_1 + \mathrm{Cos}(\ \alpha\ ) * \mathrm{Cos}(\ \beta\ ) * z_1$$
$$\phi = \mathrm{Atan2}(\ y_2,\ x_2\ ) * 180 \div \pi$$
$$\theta = \mathrm{Asin}(\ z_2\ ) * 180 \div \pi$$

D.3.42 Regional nesting SEI message semantics

The regional nesting SEI message provides a mechanism to associate SEI messages with regions of the picture. The associated SEI messages are conveyed within the regional nesting SEI message.

A regional nesting SEI message contains one or more SEI messages. When an SEI message is contained in a regional nesting SEI message, the contained SEI message is referred to as a region-nested SEI message. When an SEI message is not contained in a regional nesting SEI message, the SEI message is referred to as a non-region-nested SEI message.

For each region-nested SEI message in a regional nesting SEI message, one or more regions are specified in the regional nesting SEI message, and the semantics of the region-nested SEI message are to be interpreted as applying to each of these regions.

The list listOfRegionNestableMessageTypes includes the following types of SEI messages:

– Film grain characteristics SEI message,

– Post filter hint SEI message,

– Tone mapping information SEI message identified with a particular value of tone_map_id,

– Chroma resampling filter hint SEI message,

– Knee function information SEI message identified with a particular value of knee_function_id,

– Colour remapping information SEI message identified with a particular value of colour_remap_id,

– Content colour volume SEI message.

NOTE 1 – SEI messages of each of the following are considered different types of SEI messages: 1) tone mapping information SEI messages with different values of tone_map_id, 2) knee function information SEI messages with different values of knee_function_id, and 3) colour remapping information SEI messages with different values of colour_remap_id.

When an SEI message of a particular type in listOfRegionNestableMessageTypes has film_grain_characteristics_cancel_flag, tone_map_cancel_flag, knee_function_cancel_flag, or colour_remap_cancel_flag equal to 1, regardless of whether it is region-nested or non-region-nested, it cancels the persistence of all the region-nested SEI messages of that type, regardless of their associated regions. When an SEI message of a particular type having film_grain_characteristics_persistence_flag, tone_map_persistence_flag, knee_function_persistence_flag, or colour_remap_persistence_flag equal to 1 is region-nested, the persistence of the SEI message is determined by the semantics of the SEI message, irrespective of which region it applies to.

NOTE 2 – A region-nested SEI message has the same persistence scope as if the SEI message was non-region-nested.

NOTE 3 – A region-nested SEI message does not cancel the persistence of a non-region-nested SEI message of the same type.

The list listOfAllowedRegionalNestableMessageTypes includes all the entries in the list listOfRegionNestableMessageTypes and also the following additional types of SEI messages:

– User data registered by Rec. ITU-T T.35 SEI message,

– User data unregistered SEI message.

In bitstreams conforming to this version of this Specification, the regional nesting SEI message shall not contain any SEI message that is not in listOfAllowedRegionNestableMessageTypes. Decoders encountering a region-nested SEI message that does not belong to listOfAllowedRegionNestableMessageTypes shall ignore the region-nested SEI message.

When an access unit contains both region-nested SEI messages of a particular type in listOfRegionNestableMessageTypes and non-region-nested SEI messages of the same type, decoders shall ignore either all the region-nested SEI message of that type or all the non-region-nested SEI messages of that type. Unless indicated otherwise by some means not specified in this Specification, when an access unit contains both region-nested SEI messages of a particular type in listOfRegionNestableMessageTypes and non-region-nested SEI messages of the same type, the region-nested SEI messages should be preferred to be considered as applicable to the access unit.

A region-nested SEI messages should not be extracted and sent as a non-region-nested SEI message, as the values signalled in the region-nested SEI message may not be applicable outside the indicated regions.

regional_nesting_id contains an identifying number that may be used to identify the purpose of the one or more SEI messages that are region-nested in the regional nesting SEI message. The value of regional_nesting_id shall be in the range of 0 to $2^{16} - 1$, inclusive.

Values of regional_nesting_id from 0 to 255, inclusive, and from 512 to $2^{15} - 1$, inclusive, may be used as determined by the application. Values of regional_nesting_id from 256 to 511, inclusive, and from 2^{15} to $2^{16} - 1$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of regional_nesting_id in the range of 256 to 511, inclusive, or in the range of 2^{15} to $2^{16} - 1$, inclusive, shall ignore it.

regional_nesting_num_rect_regions specifies the number of rectangular regions specified in the regional nesting SEI message. The value of regional_nesting_num_rect_regions shall be in the range of 1 to 255, inclusive. The value of regional_nesting_num_rect_regions equal to 0 is reserved for future use by ITU-T | ISO/IEC and shall not be used in bitstreams conforming to this version of this Specification. Decoders shall ignore regional nesting SEI messages with regional_nesting_num_rect_regions equal to 0.

regional_nesting_rect_region_id[i] specifies the identifier for the i-th rectangular region specified in the regional nesting SEI message.

Unless indicated otherwise by some means not specified in this Specification, when a sample belongs to more than one region indicated as applying to more than one region-nested SEI message of a particular type in listOfRegionNestableMessageTypes, among these region-nested SEI messages, only those that are associated with the region that has the greatest value of regional_nesting_rect_region_id[ ] are considered as applying to the sample, while the rest of these region-nested SEI messages are considered as not applying to the sample.

NOTE 4 – When there are more than one of these region-nested SEI messages associated with the region that has the greatest value of regional_nesting_rect_region_id[ ], they are identical per other expressed constraints.

It is a requirement of bitstream conformance that the value of regional_nesting_rect_region_id[i] shall not be the same for any two different values of i in the range of 0 to regional_nesting_num_rect_regions − 1, inclusive, in the regional nesting SEI message.

When a region-nested SEI message of a particular type in listOfRegionNestableMessageTypes is indicated as applying to a list of regions listA in the current picture and another region-nested SEI message of the same type is indicated as applying to another list of regions listB in the current picture, it is a requirement of bitstream conformance that, for any pair of regions formed by choosing one from listA and the other from listB, the value of regional_nesting_rect_region_id[ ] of the two regions shall not be the same unless the two regions are identical (i.e., both position and size are the same) and the two region-nested SEI messages are identical.

regional_nesting_rect_left_offset[i], **regional_nesting_rect_right_offset**[i], **regional_nesting_rect_top_offset**[i], and **regional_nesting_rect_bottom_offset**[i] specify the coordinates of the i-th rectangular region specified in the SEI message. The offsets for the rectangular region are specified in units of luma samples. The i-th rectangular region contains the luma samples with horizontal picture coordinates from SubWidthC * regional_nesting_rect_left_offset[i] to pic_width_in_luma_samples − (SubWidthC * regional_nesting_rect_right_offset[i] + 1), inclusive, and vertical picture coordinates from SubHeightC * regional_nesting_rect_top_offset[i] to pic_height_in_luma_samples − (SubHeightC * regional_nesting_rect_bottom_offset[i] + 1), inclusive.

The value of SubWidthC * (regional_nesting_rect_left_offset[i] + regional_nesting_rect_right_offset[i]) shall be less than pic_width_in_luma_samples and the value of SubHeightC * (regional_nesting_rect_top_offset[i] + regional_nesting_rect_bottom_offset[i]) shall be less than pic_height_in_luma_samples.

num_sei_messages_in_regional_nesting_minus1 plus 1 specifies the number of region-nested SEI messages in the regional nesting SEI message. The value of num_sei_messages_in_regional_nesting_minus1 shall be in the range of 0 to 255, inclusive.

num_regions_for_sei_message[i] specifies the number of regions to which the i-th region-nested SEI message is associated. When regional_nesting_num_rect_regions is greater than 0, the value of num_regions_for_sei_message[i] shall be in the range of 0 to regional_nesting_num_rect_regions, inclusive. When num_regions_for_sei_message[i] is equal to 0, the i-th region-nested SEI message applies to all the regions specified in the regional nesting SEI message.

regional_nesting_sei_region_idx[i][j] specifies the index, into the list of regions specified in the regional nesting SEI message, of the j-th region to which the i-th region-nested SEI message in the regional nesting SEI message is associated. The value of regional_nesting_sei_region_idx[i][j] shall be in the range of 0 to regional_nesting_num_rect_region − 1, inclusive.

D.3.43 Motion-constrained tile sets extraction information sets SEI message semantics

The motion-constrained tile sets extraction information sets SEI message provides supplemental information that can be used in the motion-constrained tile set (MCTS) sub-bitstream extraction as specified below to generate a conforming bitstream for an MCTS set. The information consists of a number of extraction information sets, each defining a number of MCTS sets and containing RBSP bytes of the replacement VPSs, SPSs, and PPSs to be used during the MCTS sub-bitstream extraction process. Each extraction information set can be shared by multiple MCTS sets, i.e., is used for extraction of any of these MCTS sets.

An MCTS extraction information sets SEI message shall not be present in an access unit unless there is a temporal MCTS SEI message present in the access unit. A temporal MCTS SEI message present in the same access unit as an MCTS extraction information sets SEI message is referred to as the associated temporal MCTS SEI message of the MCTS extraction information sets SEI message.

An MCTS extraction information sets SEI message applies to the same set of pictures as the associated temporal MCTS SEI message, i.e., the associatedPicSet of the MCTS extraction information sets SEI message is the same as the associatedPicSet of the associated temporal MCTS SEI message.

When more than one MCTS extraction information sets SEI message is present for the pictures of associatedPicSet, these MCTS extraction information sets SEI messages shall contain identical content.

NAL units that contain tiles belonging to any particular MCTS mctsA shall not contain tiles that do not belong to mctsA.

num_info_sets_minus1 plus 1 specifies the number of extraction information sets contained in the MCTS extraction information sets SEI message. The value of num_info_sets_minus1 shall be in the range of 0 to 2047, inclusive.

The i-th extraction information set is assigned an MCTS extraction information set identifier value equal to i.

num_mcts_sets_minus1[i] plus 1 specifies the number of MCTS sets that share the i-th extraction information set. The value of num_mcts_sets_minus1[i] shall be in the range of 0 to 2047, inclusive.

num_mcts_in_set_minus1[i][j] plus 1 specifies the number of MCTSs in the j-th MCTS set that is associated with the i-th extraction information set. The value of num_mcts_in_set_minus1[i][j] shall be in the range of 0 to 511, inclusive.

idx_of_mcts_in_set[i][j][k] specifies the MCTS index of the k-th MCTS in the j-th MCTS set that is associated with the i-th extraction information set. The value of idx_of_mcts_in_set[i][j][k] shall be in the range of 0 to 511, inclusive.

slice_reordering_enabled_flag[i] equal to 1 specifies that the MCTS sub-bitstream extraction using the i-th extraction information set includes reordering of extracted slice segments and that the slice_segment_address of the j-th slice segment in bitstream order associated with any of the extracted MCTS sets with j in the range of 0 to num_slice_segments_minus1[i] is set to output_slice_segment_address[i][j]. slice_reordering_enabled_flag[i] equal to 0 indicates that the MCTS sub-bitstream extraction using the i-th extraction information set does not include a reordering of extracted slice segments and that the slice_segment_address of all extracted slice segments is calculated during extraction.

num_slice_segments_minus1[i] plus 1 specifies the number of slice segments associated with any MCTS set of the i-th extraction information set when slice_reordering_enabled_flag[i] is equal to 1. The value of num_slice_segments_minus1[i] shall be in the range of 0 to 1 024, inclusive.

output_slice_segment_address[i][j] specifies the slice segment address of the j-th slice segment in bitstream order associated with any of the MCTS sets of the i-th extraction information set when slice_reordering_enabled_flag[i] is equal to 1. The length of the output_slice_segment_address[i][j] syntax element is Ceil(Log2(PicSizeInCtbsY)) bits. The value of output_slice_segment_address shall be in the range of 0 to PicSizeInCtbsY − 1, inclusive and no value of output_slice_segment_address[i][j] shall be equal to some output_slice_segment_address[i][k] for j not equal to k in the extraction information set SEI message.

num_vps_in_info_set_minus1[i] plus 1 specifies the number of replacement VPSs in the i-th extraction information set. The value of num_vps_in_info_set_minus1[i] shall be in the range of 0 to 15, inclusive.

vps_rbsp_data_length[i][j] specifies the number of RBSP data bytes of the j-th replacement VPS in the i-th extraction information set.

num_sps_in_info_set_minus1[i] plus 1 specifies the number of replacement SPSs in the i-th extraction information set. The value of num_sps_in_info_set_minus1[i] shall be in the range of 0 to 15, inclusive.

sps_rbsp_data_length[i][j] specifies the number of RBSP data bytes of the j-th replacement SPS in the i-th extraction information set.

num_pps_in_info_set_minus1[i] plus 1 specifies the number of replacement PPSs in the i-th extraction information set. The value of num_pps_in_info_set_minus1[i] shall be in the range of 0 to 63, inclusive.

pps_nuh_temporal_id_plus1[i][j] minus1 specifies the temporal identifier of the j-th replacement PPS in the i-th extraction information set.

pps_rbsp_data_length[i][j] specifies the number of RBSP data bytes of the j-th replacement PPS in the i-th extraction information set.

mcts_alignment_bit_equal_to_zero shall be equal to 0.

vps_rbsp_data_byte[i][j][k] contains the k-th byte of the RBSP data of the j-th replacement VPS in the i-th extraction information set.

sps_rbsp_data_byte[i][j][k] contains the k-th byte of the RBSP data of the j-th replacement SPS in the i-th extraction information set.

pps_rbsp_data_byte[i][j][k] contains the k-th byte of the RBSP data of the j-th replacement PPS in the i-th extraction information set.

The MCTS sub-bitstream extraction process is specified as follows:

- Let a bitstream inBitstream, a target MCTS set index mctsSetIdxTarget, a target MCTS extraction information set identifier mctsEisIdTarget, and a target highest TemporalId value mctsTidTarget be the inputs to the MCTS sub-bitstream extraction process.
- The output of the MCTS sub-bitstream extraction process is a sub-bitstream outBitstream derived as follows:
  - The bitstream outBitstream is set to be identical to the bitstream inBitstream.
  - The lists ausWithVps, ausWithSps. and ausWithPps are set to consist of all access units within outBitstream containing non-VCL NAL units with nal_unit_type equal to VPS_NUT, SPS_NUT, or PPS_NUT.
  - Remove all SEI NAL units that contain non-MCTS-nested SEI messages.

    NOTE 1 – A "smart" bitstream extractor might include appropriate non-MCTS-nested SEI messages in the extracted MCTS sub-bitstream, provided that the SEI messages applicable to the MCTS sub-bitstream were present as MCTS-nested SEI messages in the original bitstream.
  - Remove from outBitstream all of the following NAL units:
    - VCL NAL units that contain tiles not belonging to any of the MCTSs with MCTS index equal to idx_of_mcts_in_set[mctsEisIdTarget][mctsSetIdxTarget][k] for each value of k in the range of 0 to num_mcts_in_set_minus1[mctsEisIdTarget][mctsSetIdxTarget], inclusive.
    - Non-VCL NAL units with nal_unit_type equal to VPS_NUT, SPS_NUT, or PPS_NUT.
  - Insert into each access unit within the list ausWithVps in outBitstream num_vps_in_info_set_minus1[mctsEisIdTarget] plus 1 VPS NAL units generated from the RBSP data of the list of replacement VPSs in the mctsEisIdTarget-th MCTS extraction information set. For each VPS NAL unit that is generated the nuh_layer_id is set equal to 0 and nuh_temporal_id_plus1 is set equal to 1.
  - Insert into each access unit within the list ausWithSps in outBitstream num_sps_in_info_set_minus1[mctsEisIdTarget] plus 1 SPS NAL units generated from the RBSP data of the list of replacement SPSs in the mctsEisIdTarget-th MCTS extraction information set. For each SPS NAL unit that is generated the nuh_layer_id is set equal to 0 and nuh_temporal_id_plus1 is set equal to 1.
  - Insert into each access unit within the list ausWithPps in outBitstream PPS NAL units generated from the RBSP data of the replacement PPSs, in the mctsEisIdTarget-th MCTS extraction information set, for which pps_nuh_temporal_id_plus1[mctsEisIdTarget][j] is less than or equal to mctsTidTarget. For each PPS NAL unit that is generated the nuh_layer_id is set equal to 0, and for the PPS NAL unit that is generated from the RBSP data of the j-th replacement PPS in the mctsEisIdTarget-th MCTS extraction information set, nuh_temporal_id_plus1 is set equal to pps_nuh_temporal_id_plus1[mctsEisIdTarget][j].

NOTE 2 – The values of pps_pic_parameter_set_id of the replacement PPSs should be identical to the values of pps_pic_parameter_set_id of the removed PPSs to retain the value of slice_pic_parameter_set_id in slice headers of the original bitstream.

- Remove from outBitstream all NAL units with TemporalId greater than mctsTidTarget.
- If slice_reordering_enabled_flag[mctsEISIdTarget] is equal to 0, the CTB raster and tile scanning conversion process as specified in clause 6.5.1 is invoked with the syntax element values of the replacement SPS and PPS as inputs. The output CtbAddrRsToTs[ctbAddrRs] is assigned to extCtbAddrRsToTs[ctbAddrRs] and CtbAddrTsToRs[ctbAddrTs] is assigned to extCtbAddrTsToRs[ctbAddrTs]. For each remaining VCL NAL units in outBitstream, adjust the slice segment header as follows:
  - For the first VCL NAL unit within each access unit, set the value of first_slice_segment_in_pic_flag equal to 1, and set the value of slice_segment_address to be equal to 0.
  - For each remaining VCL NAL units in outBitstream, let ctbAddrRs be the value of the raster scan address of the last CTB in the previous VCL NAL unit in bitstream order within a coded picture of outBitstream, set the value of first_slice_segment_in_pic_flag equal to 0, and set the value of slice_segment_address equal to extCtbAddrTsToRs[extCtbAddrRsToTs[ctbAddrRs] + 1].
- Otherwise (slice_reordering_enabled_flag[mctsEISIdTarget] is equal to 1), the following applies:
  - For the k-th VCL NAL units of each access unit in outBitstream, set the value of first_slice_segment_in_pic_flag equal to 0 and slice_segment_address equal to output_slice_segement_address[mctsEISIdTarget][j], where j is in the range of 0 to num_slice_segments_minus1[mctsEisIdTarget], inclusive.
  - Reorder the VCL NAL units within each access unit for ascending values of slice_segment_address.
  - For the first VCL NAL unit within each access unit, set the value of first_slice_segment_in_pic_flag equal to 1.

NOTE 3 – The extracted MCTS sub-bitstream might have a smaller luma picture size compared to the original bitstream which might require adjustment of the length of slice_segment_address and the byte_alignment( ) in slice headers.

It is a requirement of bitstream conformance for the input bitstream that any output sub-bitstream that is the output of the MCTS sub-bitstream extraction process specified in this clause shall be a conforming bitstream.

D.3.44 Motion-constrained tile sets extraction information nesting SEI message semantics

The motion-constrained tile sets extraction information nesting SEI message, also referred to as the MCTS nesting SEI message, provides a mechanism to carry and associate the SEI messages with bitstream subsets corresponding to one or more MCTSs. An SEI message contained in an MCTS nesting SEI message is referred to as MCTS-nested or an MCTS-nested SEI message, and an SEI message that is not contained in an MCTS nesting SEI message is referred to as a non-MCTS-nested SEI message.

In the MCTS sub-bitstream extraction process as specified in the semantics of the MCTS extraction information sets SEI message, the MCTS-nested SEI messages applicable to the MCTSs in an MCTS set in an access unit can be included in the corresponding access unit of the extracted sub-bitstream as non-MCTS-nested SEI messages.

An MCTS-nesting SEI message shall not be present for a picture unless the picture belongs to the associatedPicSet of a temporal MCTS SEI message. This temporal MCTS SEI message is referred to as the associated temporal MCTS SEI message of the MCTS-nesting SEI message.

An SEI NAL unit containing an MCTS nesting SEI message shall not contain any other SEI message that is not MCTS-nested in the MCTS nesting SEI message.

all_mcts_flag equal to 0 specifies that the list nesting_mctsid is set to consist of the MCTS indices indicated by all instances of idx_of_associated_mcts[i] for i from 0 to num_associated_mcts_minus1, inclusive. all_mcts_flag equal to 1 specifies that the list nesting_mctsid consists of the MCTS indices of all the MCTSs indicated by the associated temporal MCTS SEI message.

The MCTS-nested SEI messages apply to all MCTSs for which the MCTS indices are included in the list nesting_mctsid.

num_associated_mcts_minus1 plus 1 specifies the number of the following MCTS indices. The value of num_associated_mcts_minus1 shall be in the range of 0 to 511, inclusive.

idx_of_associated_mcts[i] indicates the MCTS index of the i-th MCTS associated with the following MCTS-nested SEI messages. The value of idx_of_associated_mcts[i] shall be in the range of 0 to 511, inclusive.

num_sei_messages_in_mcts_extraction_nesting_minus1 plus 1 specifies the number of the following MCTS-nested SEI messages.

mcts_nesting_zero_bit shall be equal to 0.

D.3.45 SEI manifest SEI message semantics

The SEI manifest SEI message conveys information on SEI messages that are indicated as expected (i.e., likely) to be present or not present. Such information may include the following:

– The indication that certain types of SEI messages are expected (i.e., likely) to be present (although not guaranteed to be present) in the CVS.

– For each type of SEI message that is indicated as expected (i.e., likely) to be present in the CVS, the degree of expressed necessity of interpretation of the SEI messages of this type, as follows:

  – The degree of necessity of interpretation of an SEI message type may be indicated as "necessary", "unnecessary", or "undetermined".

  – An SEI message is indicated by the encoder (i.e., the content producer) as being "necessary" when the information conveyed by the SEI message is considered as necessary for interpretation by the decoder or receiving system in order to properly process the content and enable an adequate user experience; it does not mean that the bitstream is required to contain the SEI message in order to be a conforming bitstream. It is at the discretion of the encoder to determine which SEI messages are to be considered as necessary in a particular CVS. However, it is suggested that some SEI messages, such as the frame packing arrangement, segmented rectangular frame packing arrangement, and omnidirectional projection indication SEI messages, should typically be considered as necessary.

– The indication that certain types of SEI messages are expected (i.e., likely) not to be present (although not guaranteed not to be present) in the CVS.

NOTE – An example of such a usage of an SEI manifest SEI message is to express the expectation that there are no frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, display orientation SEI messages, or omnidirectional projection indication SEI messages in the CVS, and therefore that the rendering of the decoded video pictures for display purposes would not need any of the additional post-processing that is commonly associated with the interpretation of these SEI messages.

The content of an SEI manifest SEI message may, for example, be used by transport-layer or systems-layer processing elements to determine whether the CVS is suitable for delivery to a receiving and decoding system, based on whether the receiving system can properly process the CVS to enable an adequate user experience or whether the CVS satisfies the application needs.

When an SEI manifest SEI message is present in any access unit of a CVS, an SEI manifest SEI message shall be present in the first access unit of the CVS. The SEI manifest SEI message persists in decoding order from the current access unit until the end of the CVS. When there are multiple SEI manifest SEI messages present in a CVS, they shall have the same content.

An SEI NAL unit containing an SEI manifest SEI message shall not contain any other SEI messages other than SEI prefix indication SEI messages. When present in an SEI NAL unit, the SEI manifest SEI message shall be the first SEI message in the SEI NAL unit.

manifest_num_sei_msg_types specifies the number of types of SEI messages for which information is provided in the SEI manifest SEI message.

manifest_sei_payload_type[i] indicates the payloadType value of the i-th type of SEI message for which information is provided in the SEI manifest SEI message. The values of manifest_sei_payload_type[m] and manifest_sei_payload_type[n] shall not be identical when m is not equal to n.

manifest_sei_description[i] provides information on SEI messages with payloadType equal to manifest_sei_payload_type[i] as specified in Table D.23.

Table D.23 – Interpretation of manifest_sei_description[i]

Value	Description
0	Indicates that there is no SEI message with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS.
1	Indicates that there are SEI messages with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS, and these SEI messages are considered as necessary.
2	Indicates that there are SEI messages with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS, and these SEI messages are considered as unnecessary.

3	Indicates that there are SEI messages with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS, and the necessity of these SEI messages is undetermined.
4..255	Reserved

The value of manifest_sei_description[i] shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this version of this Specification. Other values for manifest_sei_description[i] are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of manifest_sei_description[i] greater than or equal to 4 to appear in the syntax and shall ignore all information for payloadType equal to manifest_sei_payload_type[i] signalled in the SEI manifest SEI message and shall ignore all SEI prefix indication SEI messages with prefix_sei_payload_type equal to manifest_sei_payload_type[i] when manifest_sei_description[i] is greater than or equal to 4.

D.3.46 SEI prefix indication SEI message semantics

The SEI prefix indication SEI message carries one or more SEI prefix indications for SEI messages of a particular value of payloadType. Each SEI prefix indication is a bit string that follows the SEI payload syntax of that value of payloadType and contains a number of complete syntax elements starting from the first syntax element in the SEI payload.

Each SEI prefix indication for an SEI message of a particular value of payloadType indicates that one or more SEI messages of this value of payloadType are expected (i.e., likely) to be present in the CVS and to start with the provided bit string. A starting bit string would typically contain only a true subset of an SEI payload of the type of SEI message indicated by the payloadType, may contain a complete SEI payload, and shall not contain more than a complete SEI payload. It is not prohibited for SEI messages of the indicated value of payloadType to be present that do not start with any of the indicated bit strings.

These SEI prefix indications should provide sufficient information for indicating what type of processing is needed or what type of content is included. The former (type of processing) indicates decoder-side processing capability, e.g., whether some type of frame unpacking is needed. The latter (type of content) indicates, for example, whether the bitstream contains subtitle captions in a particular language.

The content of an SEI prefix indication SEI message may, for example, be used by transport-layer or systems-layer processing elements to determine whether the CVS is suitable for delivery to a receiving and decoding system, based on whether the receiving system can properly process the CVS to enable an adequate user experience or whether the CVS satisfies the application needs (as determined in some manner by external means outside the scope of this Specification).

In one example, when the payloadType indicates the frame packing arrangement SEI message, an SEI prefix indication should include up to at least the syntax element frame_packing_arrangement_type; and when the payloadType indicates the omnidirectional projection indication SEI message, an SEI prefix indication should include up to at least the syntax element projection_type.

In another example, for user data registered SEI messages that are used to carry captioning information, an SEI prefix indication should include up to at least the language code; and for user data unregistered SEI messages extended for private use, an SEI prefix indication should include up to at least the UUID.

When an SEI prefix indication SEI message is present in any access unit of a CVS, an SEI prefix indication SEI message shall be present in the first access unit of the CVS. The SEI prefix indication SEI message persists in decoding order from the current access unit until the end of the CVS. When there are multiple SEI prefix indication SEI messages present in a CVS for a particular value of payloadType, they shall have the same content.

An SEI NAL unit containing an SEI prefix indication SEI message for a particular value of payloadType shall not contain any other SEI messages other than an SEI manifest SEI message and SEI prefix indication SEI messages for other values of payloadType.

prefix_sei_payload_type indicates the payloadType value of the SEI messages for which one or more SEI prefix indications are provided in the SEI prefix indication SEI message. When an SEI manifest SEI message is also present for the CVS, the value of prefix_sei_payload_type shall be equal to one of the manifest_sei_payload_type[m] values for which manifest_sei_description[m] is equal to 1 to 3, inclusive, as indicated by an SEI manifest SEI message that applies to the CVS.

num_sei_prefix_indications_minus1 plus 1 specifies the number of SEI prefix indications.

num_bits_in_prefix_indication_minus1[i] plus 1 specifies the number of bits in the i-th SEI prefix indication.

sei_prefix_data_bit[i][j] specifies the j-th bit of the i-th SEI prefix indication.

The bits sei_prefix_data_bit[i][j] for j ranging from 0 to num_bits_in_prefix_indication_minus1[i], inclusive, follow the syntax of the SEI payload with payloadType equal to prefix_sei_payload_type, and contain a number of complete

syntax elements starting from the first syntax element in the SEI payload syntax, and may or may not contain all the syntax elements in the SEI payload syntax. The last bit of these bits (i.e., the bit sei_prefix_data_bit[i][num_bits_in_prefix_indication_minus1[i]]) shall be the last bit of a syntax element in the SEI payload syntax, unless it is a bit within an itu_t_t35_payload_byte or user_data_payload_byte.

NOTE – The exception for itu_t_t35_payload_byte and user_data_payload_byte is provided because these syntax elements may contain externally-specified syntax elements, and the determination of the boundaries of such externally-specified syntax elements is a matter outside the scope of this Specification.

byte_alignment_bit_equal_to_one shall be equal to 1.

D.3.47 Annotated regions SEI message semantics

The annotated regions SEI message carries parameters that identify annotated regions using bounding boxes representing the size and location of identified objects.

ar_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous annotated regions SEI message that is associated with one or more layers to which the annotated regions SEI message applies. ar_cancel_flag equal to 0 indicates that annotated regions information follows.

When ar_cancel_flag equal to 1 or a new CLVS of the current layer begins, the variables LabelAssigned[i], ObjectTracked[i], and ObjectBoundingBoxAvail are set equal to 0 for i in the range of 0 to 255, inclusive.

Let picA be the current picture. Each region identified in the annotated regions SEI message persists for the current layer in output order until any of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing an annotated regions SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, and the semantics of the annotated regions SEI message for PicB cancels the persistence of the region identified in the annotated regions SEI message for PicA.

ar_not_optimized_for_viewing_flag equal to 1 indicates that the decoded pictures that the annotated regions SEI message applies to are not optimized for user viewing, but rather are optimized for some other purpose such as algorithmic object classification performance. ar_not_optimized_for_viewing_flag equal to 0 indicates that the decoded pictures that the annotated regions SEI message applies to may or may not be optimized for user viewing.

ar_true_motion_flag equal to 1 indicates that the motion information in the coded pictures that the annotated regions SEI message applies to was selected with a goal of accurately representing object motion for objects in the annotated regions. ar_true_motion_flag equal to 0 indicates that the motion information in the coded pictures that the annotated regions SEI message applies to may or may not be selected with a goal of accurately representing object motion for objects in the annotated regions.

ar_occluded_object_flag equal to 1 indicates that the ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]], and ar_bounding_box_height[ar_object_idx[i]] syntax elements represent the size and location of an object or a portion of an object that may not be visible or may be only partially visible within the cropped decoded picture. ar_occluded_object_flag equal to 0 indicates that the ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]], and ar_bounding_box_height[ar_object_idx[i]] syntax elements represent the size and location of an object that is entirely visible within the cropped decoded picture. It is a requirement of bitstream conformance that the value of ar_occluded_object_flag shall be the same for all annotated_regions( ) syntax structures within a CLVS.

ar_partial_object_flag_present_flag equal to 1 indicates that ar_partial_object_flag[ar_object_idx[i]] syntax elements are present. ar_partial_object_flag_present_flag equal to 0 indicates that ar_partial_object_flag[ar_object_idx[i]] syntax elements are not present. It is a requirement of bitstream conformance that the value of ar_partial_object_flag_present_flag shall be the same for all annotated_regions( ) syntax structures within a CLVS.

ar_object_label_present_flag equal to 1 indicates that label information corresponding to objects in the annotated regions is present. ar_object_label_present_flag equal to 0 indicates that label information corresponding to the objects in the annotated regions is not present.

ar_object_confidence_info_present_flag equal to 1 indicates that ar_object_confidence[ar_object_idx[i]] syntax elements are present. ar_object_confidence_info_present_flag equal to 0 indicates that ar_object_confidence[ar_object_idx[i]] syntax elements are not present. It is a requirement of bitstream conformance that the value of ar_object_confidence_present_flag shall be the same for all annotated_regions( ) syntax structures within a CLVS.

ar_object_confidence_length_minus1 plus 1 specifies the length, in bits, of the ar_object_confidence[ar_object_idx[i]] syntax elements. It is a requirement of bitstream conformance that the value of ar_object_confidence_length_minus1 shall be the same for all annotated_regions( ) syntax structures within a CLVS.

ar_object_label_language_present_flag equal to 1 indicates that the ar_object_label_language syntax element is present. ar_object_label_language_present_flag equal to 0 indicates that the ar_object_label_language syntax element is not present.

ar_bit_equal_to_zero shall be equal to zero.

ar_object_label_language contains a language tag as specified by IETF RFC 5646 followed by a null termination byte equal to 0x00. The length of the ar_object_label_language syntax element shall be less than or equal to 255 bytes, not including the null termination byte. When not present, the language of the label is unspecified.

ar_num_label_updates indicates the total number of labels associated with the annotated regions that are signalled. The value of ar_num_label_updates shall be in the range of 0 to 255, inclusive.

ar_label_idx[i] indicates the index of the signalled label. The value of ar_label_idx[i] shall be in the range of 0 to 255, inclusive.

ar_label_cancel_flag equal to 1 cancels the persistence scope of the ar_label_idx[i]-th label. ar_label_cancel_flag equal to 0 indicates that the ar_label_idx[i]-th label is assigned a signalled value.

LabelAssigned[ar_label_idx[i]] equal to 1 indicates that the ar_label_idx[i]-th label is assigned. LabelAssigned[ar_label_idx[i]] equal to 0 indicates that the ar_label_idx[i]-th label is not assigned.

ar_label[ar_label_idx[i]] specifies the contents of the ar_label_idx[i] –th label. The length of the ar_label[ar_label_idx[i]] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

ar_num_object_updates indicates the number of object updates to be signalled. ar_num_object_updates shall be in the range of 0 to 255, inclusive.

ar_object_idx[i] is the index of the object parameters to be signalled. ar_object_idx[i] shall be in the range of 0 to 255, inclusive.

ar_object_cancel_flag equal to 1 cancels the persistence scope of the ar_object_idx[i]-th object. ar_object_cancel_flag equal to 0 indicates that parameters associated with the ar_object_idx[i]-th object tracked object are signalled.

ObjectTracked[ar_object_idx[i]] equal to 1 indicates that the object_idx[i]-th object is tracked. ObjectTracked[ar_object_idx[i]] equal to 0 indicates that the object_idx[i]-th object is not tracked.

ar_object_label_update_flag equal to 1 indicates that an object label is signalled. ar_object_label_update_flag equal to 0 indicates that an object label is not signalled.

ar_object_label_idx[ar_object_idx[i]] indicates the index of the label corresponding to the ar_object_idx[i]-th object. When ar_object_label_idx[ar_object_idx[i]] is not present, its value is inferred from a previous annotated regions SEI message in output order in the same CLVS, if any. The value of ar_object_label_idx[ar_object_idx[i]] shall be in the range of 0 to 255, inclusive.

ar_bounding_box_update_flag equal to 1 indicates that object bounding box parameters are signalled. ar_bounding_box_update_flag equal to 0 indicates that object bounding box parameters are not signalled.

ar_bounding_box_cancel_flag equal to 1 cancels the persistence scope of the ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]], ar_bounding_box_height[ar_object_idx[i]]. ar_partial_object_flag[ar_object_idx[i]], and ar_object_confidence[ar_object_idx[i]]. ar_bounding_box_cancel_flag equal to 0 indicates that ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]] ar_bounding_box_height[ar_object_idx[i]] ar_partial_object_flag[ar_object_idx[i]], and ar_object_confidence[ar_object_idx[i]] syntax elements are signalled.

ObjectBoundingBoxAvail[ar_object_idx[i]] equal to 1 indicates that the bounding box information of the object_idx[i]-th object is signalled. ObjectBoundingBoxAvail[ar_object_idx[i]] equal to 0 indicates that the bounding box information of the object_idx[i]-th object is not signalled.

ar_bounding_box_top[ar_object_idx[i]], **ar_bounding_box_left**[ar_object_idx[i]], **ar_bounding_box_width**[ar_object_idx[i]], and **ar_bounding_box_height**[ar_object_idx[i]] specify the coordinates of the top-left corner and the width and height, respectively, of the bounding box of the ar_object_idx[i]-th object in the cropped decoded picture, relative to the conformance cropping window specified by the active SPS.

Let croppedWidth and croppedHeight be the width and height, respectively, of the cropped decoded picture in units of luma samples, as specified by Equations D-28 and D-29.

The value of ar_bounding_box_left[ar_object_idx[i]] shall be in the range of 0 to croppedWidth / SubWidthC − 1, inclusive.

The value of ar_bounding_box_top[ar_object_idx[i]] shall be in the range of 0 to croppedHeight / SubHeightC − 1, inclusive.

The value of ar_bounding_box_width[ar_object_idx[i]] shall be in the range of 0 to croppedWidth / SubWidthtC − ar_bounding_box_left[ar_object_idx[i]], inclusive.

The value of ar_bounding_box_height[ar_object_idx[i]] shall be in the range of 0 to croppedHeight / SubHeightC − ar_bounding_box_top[ar_object_idx[i]], inclusive.

The identified object rectangle contains the luma samples with horizontal picture coordinates from SubWidthC * (conf_win_left_offset + ar_bounding_box_left[ar_object_idx[i]]) to SubWidthC * (conf_win_left_offset + ar_bounding_box_left[ar_object_idx[i]] + ar_bounding_box_width[ar_object_idx[i]]) − 1, inclusive, and vertical picture coordinates from SubHeightC * (conf_win_top_offset + ar_bounding_box_top[ar_object_idx[i]]) to SubHeightC * (conf_win_top_offset + ar_bounding_box_top[ar_object_idx[i]] + ar_bounding_box_height[ar_object_idx[i]]) − 1, inclusive.

When ChromaArrayType is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates (x / SubWidthC, y / SubHeightC), where (x, y) are the picture coordinates of the specified luma samples.

The values of ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]] and ar_bounding_box_height[ar_object_idx[i]] persist in output order within the CLVS for each value of ar_object_idx[i]. When not present, the values of ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]] or ar_bounding_box_height[ar_object_idx[i]] are inferred from a previous annotated regions SEI message in output order in the CLVS, if any.

ar_partial_object_flag[ar_object_idx[i]] equal to 1 indicates that the ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]] and ar_bounding_box_height[ar_object_idx[i]] syntax elements represent the size and location of an object that is only partially visible within the cropped decoded picture. ar_partial_object_flag[ar_object_idx[i]] equal to 0 indicates that the ar_bounding_box_top[ar_object_idx[i]], ar_bounding_box_left[ar_object_idx[i]], ar_bounding_box_width[ar_object_idx[i]] and ar_bounding_box_height[ar_object_idx[i]] syntax elements represent the size and location of an object that may or may not be only partially visible within the cropped decoded picture. When not present, the value of ar_partial_object_flag[ar_object_idx[i]] is inferred from a previous annotated regions SEI message in output order in the CLVS, if any.

ar_object_confidence[ar_object_idx[i]] indicates the degree of confidence associated with the ar_object_idx[i]-th object, in units of $2^{-(\text{ar\_object\_confidence\_length\_minus1} + 1)}$, such that a higher value of ar_object_confidence[ar_object_idx[i]] indicates a higher degree of confidence. The length of the ar_object_confidence[ar_object_idx[i]] syntax element is ar_object_confidence_length_minus1 + 1 bits. When not present, the value of_object_confidence[ar_object_idx[i]] is inferred from a previous annotated regions SEI message in output order in the CLVS, if any.

D.3.48 Shutter interval information SEI message semantics

The shutter interval information SEI message indicates the shutter interval for the associated video source pictures prior to encoding, e.g., for camera-captured content, the shutter interval is amount of time that an image sensor is exposed to produce each source picture.

When a shutter interval information SEI message is present for any picture of a CLVS of a particular layer, a shutter interval information SEI message shall be present for the first picture of the CLVS. The shutter interval information SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All shutter interval information SEI messages that apply to the same CLVS shall have the same content.

sii_time_scale specifies the number of time units that pass in one second. The value of sii_time_scale shall be greater than 0. For example, a time coordinate system that measures time using a 27 MHz clock has an sii_time_scale of 27 000 000.

fixed_shutter_interval_within_clvs_flag equal to 1 specifies that the indicated shutter interval is the same for all temporal sub-layers in the CLVS. fixed_shutter_interval_within_clvs_flag equal to 0 specifies that the indicated shutter interval may not be the same for all temporal sub-layers in the CLVS. When the value of sps_max_sub_layers_minus1 is equal to 0, the value of fixed_shutter_interval_within_clvs_flag shall be equal to 1.

sii_num_units_in_shutter_interval, when fixed_shutter_interval_within_clvs_flag is equal to 1, specifies the number of time units of a clock operating at the frequency sii_time_scale Hz that corresponds to the indicated shutter interval of each picture in the CLVS. The value 0 may be used to indicate that the associated video content contains screen capture content, computer generated content, or other non-camera-captured content.

The indicated shutter interval, denoted by the variable shutterInterval, in units of seconds, is equal to the quotient of sii_num_units_in_shutter_interval divided by sii_time_scale. For example, to represent a shutter interval equal to 0.04 seconds, sii_time_scale may be equal to 27 000 000 and sii_num_units_in_shutter_interval may be equal to 1 080 000.

sii_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CLVS referring to the SPS. The value of sii_max_sub_layers_minus1 shall be equal to the value of sps_max_sub_layers_minus1 in the SPS.

NOTE – For example, the information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of ATSC A/341:2019 Annex D when sii_max_sub_layers_minus1 is equal to 1 and fixed_shutter_interval_within_clvs_flag is equal to 0.

sub_layer_num_units_in_shutter_interval[i], when present, specifies the number of time units of a clock operating at the frequency sii_time_scale Hz that corresponds to the shutter interval of each picture in the sub-layer representation with TemporalId equal to i in the CLVS. The sub-layer shutter interval for the sub-layer representation with TemporalId equal to i, denoted by the variable subLayerShutterInterval[i], in units of seconds, is equal to the quotient of sub_layer_num_units_in_shutter_interval[i] divided by sii_time_scale.

The variable subLayerShutterInterval[i], corresponding to the indicated shutter interval of each picture in the sub-layer representation with TemporalId equal to i in the CLVS, is thus derived as follows:

```
if( fixed_shutter_interval_within_clvs_flag )
   subLayerShutterInterval[ i ] = sii_num_units_in_shutter_interval ÷ sii_time_scale          (D-63)
else
   subLayerShutterInterval[ i ] = sub_layer_num_units_in_shutter_interval[ i ] ÷ sii_time_scale
```

D.3.49 Reserved SEI message semantics

The reserved SEI message consists of data reserved for future backward-compatible use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that bitstreams shall not contain reserved SEI messages until and unless the use of such messages has been specified by ITU-T | ISO/IEC. Decoders shall ignore reserved SEI messages.

Annex E

Video usability information

(This annex forms an integral part of this Recommendation | International Standard.)

E.1 General

This annex specifies syntax and semantics of the VUI parameters of the SPSs.

VUI parameters are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Specification (see Annex C and clause F.13 for the specification of output order conformance). Some VUI parameters are required to check bitstream conformance and for output timing decoder conformance.

In this annex, specification for presence of VUI parameters is also satisfied when those parameters (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. When present in the bitstream, VUI parameters shall follow the syntax and semantics specified in this annex. When the content of VUI parameters is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the VUI parameters is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

E.2 VUI syntax

E.2.1 VUI parameters syntax

vui_parameters( ) {	**Descriptor**
aspect_ratio_info_present_flag	u(1)
if(aspect_ratio_info_present_flag) {	
aspect_ratio_idc	u(8)
if(aspect_ratio_idc = = EXTENDED_SAR) {	
sar_width	u(16)
sar_height	u(16)
}	
}	
overscan_info_present_flag	u(1)
if(overscan_info_present_flag)	
overscan_appropriate_flag	u(1)
video_signal_type_present_flag	u(1)
if(video_signal_type_present_flag) {	
video_format	u(3)
video_full_range_flag	u(1)
colour_description_present_flag	u(1)
if(colour_description_present_flag) {	
colour_primaries	u(8)
transfer_characteristics	u(8)
matrix_coeffs	u(8)
}	
}	
chroma_loc_info_present_flag	u(1)
if(chroma_loc_info_present_flag) {	
chroma_sample_loc_type_top_field	ue(v)
chroma_sample_loc_type_bottom_field	ue(v)

}	
neutral_chroma_indication_flag	u(1)
field_seq_flag	u(1)
frame_field_info_present_flag	u(1)
default_display_window_flag	u(1)
if(default_display_window_flag) {	
def_disp_win_left_offset	ue(v)
def_disp_win_right_offset	ue(v)
def_disp_win_top_offset	ue(v)
def_disp_win_bottom_offset	ue(v)
}	
vui_timing_info_present_flag	u(1)
if(vui_timing_info_present_flag) {	
vui_num_units_in_tick	u(32)
vui_time_scale	u(32)
vui_poc_proportional_to_timing_flag	u(1)
if(vui_poc_proportional_to_timing_flag)	
vui_num_ticks_poc_diff_one_minus1	ue(v)
vui_hrd_parameters_present_flag	u(1)
if(vui_hrd_parameters_present_flag)	
hrd_parameters(1, sps_max_sub_layers_minus1)	
}	
bitstream_restriction_flag	u(1)
if(bitstream_restriction_flag) {	
tiles_fixed_structure_flag	u(1)
motion_vectors_over_pic_boundaries_flag	u(1)
restricted_ref_pic_lists_flag	u(1)
min_spatial_segmentation_idc	ue(v)
max_bytes_per_pic_denom	ue(v)
max_bits_per_min_cu_denom	ue(v)
log2_max_mv_length_horizontal	ue(v)
log2_max_mv_length_vertical	ue(v)
}	
}	

E.2.2 HRD parameters syntax

hrd_parameters(commonInfPresentFlag, maxNumSubLayersMinus1) {	**Descriptor**
if(commonInfPresentFlag) {	
nal_hrd_parameters_present_flag	u(1)
vcl_hrd_parameters_present_flag	u(1)
if(nal_hrd_parameters_present_flag || vcl_hrd_parameters_present_flag) {	
sub_pic_hrd_params_present_flag	u(1)
if(sub_pic_hrd_params_present_flag) {	
tick_divisor_minus2	u(8)
du_cpb_removal_delay_increment_length_minus1	u(5)
sub_pic_cpb_params_in_pic_timing_sei_flag	u(1)
dpb_output_delay_du_length_minus1	u(5)
}	
bit_rate_scale	u(4)
cpb_size_scale	u(4)
if(sub_pic_hrd_params_present_flag)	
cpb_size_du_scale	u(4)
initial_cpb_removal_delay_length_minus1	u(5)
au_cpb_removal_delay_length_minus1	u(5)
dpb_output_delay_length_minus1	u(5)
}	
}	
for(i = 0; i <= maxNumSubLayersMinus1; i++) {	
fixed_pic_rate_general_flag[i]	u(1)
if(!fixed_pic_rate_general_flag[i])	
fixed_pic_rate_within_cvs_flag[i]	u(1)
if(fixed_pic_rate_within_cvs_flag[i])	
elemental_duration_in_tc_minus1[i]	ue(v)
else	
low_delay_hrd_flag[i]	u(1)
if(!low_delay_hrd_flag[i])	
cpb_cnt_minus1[i]	ue(v)
if(nal_hrd_parameters_present_flag)	
sub_layer_hrd_parameters(i)	
if(vcl_hrd_parameters_present_flag)	
sub_layer_hrd_parameters(i)	
}	
}	

E.2.3 Sub-layer HRD parameters syntax

sub_layer_hrd_parameters(subLayerId) {	**Descriptor**
for(i = 0; i < CpbCnt; i++) {	
bit_rate_value_minus1[i]	ue(v)
cpb_size_value_minus1[i]	ue(v)
if(sub_pic_hrd_params_present_flag) {	
cpb_size_du_value_minus1[i]	ue(v)
bit_rate_du_value_minus1[i]	ue(v)
}	
cbr_flag[i]	u(1)
}	
}	

E.3 VUI semantics

E.3.1 VUI parameters semantics

aspect_ratio_info_present_flag equal to 1 specifies that aspect_ratio_idc is present. aspect_ratio_info_present_flag equal to 0 specifies that aspect_ratio_idc is not present.

aspect_ratio_idc specifies the value of the sample aspect ratio of the luma samples. Table E.1 shows the meaning of the code. When aspect_ratio_idc indicates EXTENDED_SAR, the sample aspect ratio is represented by sar_width: sar_height. When the aspect_ratio_idc syntax element is not present, the value of aspect_ratio_idc is inferred to be equal to 0. Values of aspect_ratio_idc in the range of 17 to 254, inclusive, are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret values of aspect_ratio_idc in the range of 17 to 254, inclusive, as equivalent to the value 0.

Table E.1 – Interpretation of sample aspect ratio indicator

aspect_ratio_idc	Sample aspect ratio	Examples of use (informative)
0	Unspecified	
1	1:1 ("square")	7680x4320 16:9 frame without horizontal overscan 3840x2160 16:9 frame without horizontal overscan 1280x720 16:9 frame without horizontal overscan 1920x1080 16:9 frame without horizontal overscan (cropped from 1920x1088) 640x480 4:3 frame without horizontal overscan
2	12:11	720x576 4:3 frame with horizontal overscan 352x288 4:3 frame without horizontal overscan
3	10:11	720x480 4:3 frame with horizontal overscan 352x240 4:3 frame without horizontal overscan
4	16:11	720x576 16:9 frame with horizontal overscan 528x576 4:3 frame without horizontal overscan
5	40:33	720x480 16:9 frame with horizontal overscan 528x480 4:3 frame without horizontal overscan
6	24:11	352x576 4:3 frame without horizontal overscan 480x576 16:9 frame with horizontal overscan
7	20:11	352x480 4:3 frame without horizontal overscan 480x480 16:9 frame with horizontal overscan
8	32:11	352x576 16:9 frame without horizontal overscan
9	80:33	352x480 16:9 frame without horizontal overscan
10	18:11	480x576 4:3 frame with horizontal overscan
11	15:11	480x480 4:3 frame with horizontal overscan
12	64:33	528x576 16:9 frame without horizontal overscan
13	160:99	528x480 16:9 frame without horizontal overscan
14	4:3	1440x1080 16:9 frame without horizontal overscan
15	3:2	1280x1080 16:9 frame without horizontal overscan
16	2:1	960x1080 16:9 frame without horizontal overscan
17..254	Reserved	
255	EXTENDED_SAR	

NOTE 1 – For the examples in Table E.1, the term "without horizontal overscan" refers to display processes in which the display area matches the area of the cropped decoded pictures and the term "with horizontal overscan" refers to display processes in which some parts near the left or right border of the cropped decoded pictures are not visible in the display area. As an example, the entry "720x576 4:3 frame with horizontal overscan" for aspect_ratio_idc equal to 2 refers to having an area of 704x576 luma samples (which has an aspect ratio of 4:3) of the cropped decoded frame (720x576 luma samples) that is visible in the display area.

NOTE 2 – For the examples in Table E.1, the frame spatial resolutions shown as examples of use would be the dimensions of the conformance cropping window when field_seq_flag is equal to 0 and would have twice the height of the dimensions of the conformance cropping window when field_seq_flag is equal to 1.

sar_width indicates the horizontal size of the sample aspect ratio (in arbitrary units).

sar_height indicates the vertical size of the sample aspect ratio (in the same arbitrary units as sar_width).

sar_width and sar_height shall be relatively prime or equal to 0. When aspect_ratio_idc is equal to 0 or sar_width is equal to 0 or sar_height is equal to 0, the sample aspect ratio is unspecified in this Specification.

overscan_info_present_flag equal to 1 specifies that the overscan_appropriate_flag is present. When overscan_info_present_flag is equal to 0 or is not present, the preferred display method for the video signal is unspecified.

overscan_appropriate_flag equal to 1 indicates that the cropped decoded pictures output are suitable for display using overscan. overscan_appropriate_flag equal to 0 indicates that the cropped decoded pictures output contain visually important information in the entire region out to the edges of the conformance cropping window of the picture, such that the cropped decoded pictures output should not be displayed using overscan. Instead, they should be displayed using either an exact match between the display area and the conformance cropping window, or using underscan. As used in this paragraph, the term "overscan" refers to display processes in which some parts near the borders of the cropped decoded

pictures are not visible in the display area. The term "underscan" describes display processes in which the entire cropped decoded pictures are visible in the display area, but they do not cover the entire display area. For display processes that neither use overscan nor underscan, the display area exactly matches the area of the cropped decoded pictures.

NOTE 3 – For example, overscan_appropriate_flag equal to 1 might be used for entertainment television programming, or for a live view of people in a videoconference and overscan_appropriate_flag equal to 0 might be used for computer screen capture or security camera content.

video_signal_type_present_flag equal to 1 specifies that video_format, video_full_range_flag and colour_description_present_flag are present. video_signal_type_present_flag equal to 0, specify that video_format, video_full_range_flag and colour_description_present_flag are not present.

NOTE 4 – Some of the semantics of video signal type parameters associated with video_signal_type_present_flag equal to 1 are expressed in terms of the properties of source pictures prior to operation of the encoding process, which is outside the scope of this Specification. This is partly for historical reasons and due to the common general practice of how the indicated data is typically described in industry publications. The actual intent for providing this syntax in the bitstream is to assist decoding systems to properly interpret and make effective use of the decoded video pictures, e.g., for use by the display process (which is also outside the scope of this Specification, but for which having an indication of how the pictures should be interpreted is important).

video_format indicates the representation of the pictures as specified in Table E.2, before being coded in accordance with this Specification. When the video_format syntax element is not present, the value of video_format is inferred to be equal to 5. The values 6 and 7 for video_format are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret the values 6 and 7 for video_format as equivalent to the value 5.

Table E.2 – Meaning of video_format

video_format	Meaning
0	Component
1	PAL
2	NTSC
3	SECAM
4	MAC
5	Unspecified video format

video_full_range_flag indicates the black level and range of the luma and chroma signals as derived from E'_Y, E'_{PB}, and E'_{PR} or E'_R, E'_G, and E'_B real-valued component signals.

When the video_full_range_flag syntax element is not present, the value of video_full_range_flag is inferred to be equal to 0.

colour_description_present_flag equal to 1 specifies that colour_primaries, transfer_characteristics, and matrix_coeffs are present. colour_description_present_flag equal to 0 specifies that colour_primaries, transfer_characteristics, and matrix_coeffs are not present.

colour_primaries indicates the chromaticity coordinates of the source primaries as specified in Table E.3 in terms of the CIE 1931 definition of x and y as specified in ISO 11664-1.

When the colour_primaries syntax element is not present, the value of colour_primaries is inferred to be equal to 2 (the chromaticity is unspecified or is determined by the application). Values of colour_primaries that are identified as reserved in Table E.3 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of colour_primaries as equivalent to the value 2.

Table E.3 – Colour primaries interpretation using the colour_primaries syntax element

Value	Primaries			Informative remark
0	Reserved			For future use by ITU-T | ISO/IEC
1	primary green blue red white D65	x 0.300 0.150 0.640 0.3127	y 0.600 0.060 0.330 0.3290	Rec. ITU-R BT.709-6 Rec. ITU-R BT.1361-0 conventional colour gamut system and extended colour gamut system (historical) IEC 61966-2-1 sRGB or sYCC IEC 61966-2-4 SMPTE RP 177 (1993) Annex B
2	Unspecified			Image characteristics are unknown or are determined by the application.
3	Reserved			For future use by ITU-T | ISO/IEC
4	primary green blue red white C	x 0.21 0.14 0.67 0.310	y 0.71 0.08 0.33 0.316	Rec. ITU-R BT.470-6 System M (historical) NTSC Recommendation for transmission standards for colour television (1953) FCC Title 47 Code of Federal Regulations (2003) 73.682 (a) (20)
5	primary green blue red white D65	x 0.29 0.15 0.64 0.3127	y 0.60 0.06 0.33 0.3290	Rec. ITU-R BT.470-6 System B, G (historical) Rec. ITU-R BT.601-7 625 Rec. ITU-R BT.1358-0 625 (historical) Rec. ITU-R BT.1700-0 625 PAL and 625 SECAM
6	primary green blue red white D65	x 0.310 0.155 0.630 0.3127	y 0.595 0.070 0.340 0.3290	Rec. ITU-R BT.601-7 525 Rec. ITU-R BT.1358-1 525 or 625 (historical) Rec. ITU-R BT.1700-0 NTSC SMPTE ST 170 (2004) (functionally the same as the value 7)
7	primary green blue red white D65	x 0.310 0.155 0.630 0.3127	y 0.595 0.070 0.340 0.3290	SMPTE ST 240 (1999, historical) (functionally the same as the value 6)
8	primary green blue red white C	x 0.243 0.145 0.681 0.310	y 0.692 (Wratten 58) 0.049 (Wratten 47) 0.319 (Wratten 25) 0.316	Generic film (colour filters using Illuminant C)
9	primary green blue red white D65	x 0.170 0.131 0.708 0.3127	y 0.797 0.046 0.292 0.3290	Rec. ITU-R BT.2020-2 Rec. ITU-R BT.2100-2
10	primary green (Y) blue (Z) red (X) centre white	x 0.0 0.0 1.0 1 ÷ 3	y 1.0 0.0 0.0 1 ÷ 3	SMPTE ST 428-1 (2006) (CIE 1931 XYZ)
11	primary green blue red white	x 0.265 0.150 0.680 0.314	y 0.690 0.060 0.320 0.351	SMPTE RP 431-2 (2011) SMPTE ST 2113 (2019) "P3DCI"

Table E.3 – Colour primaries interpretation using the colour_primaries syntax element

Value	Primaries			Informative remark
12	primary green blue red white D65	x 0.265 0.150 0.680 0.3127	y 0.690 0.060 0.320 0.3290	SMPTE EG 432-1 (2010) SMPTE ST 2113 (2019) "P3D65"
13..21	Reserved			For future use by ITU-T | ISO/IEC
22	primary green blue red white D65	x 0.295 0.155 0.630 0.3127	y 0.605 0.077 0.340 0.3290	EBU Tech. 3213-E (1975)
23..255	Reserved			For future use by ITU-T | ISO/IEC

transfer_characteristics, as specified in Table E.4, either indicates the reference opto-electronic transfer characteristic function of the source picture as a function of a source input linear optical intensity L_c with a nominal real-valued range of 0 to 1 or indicates the inverse of the reference electro-optical transfer characteristic function as a function of an output linear optical intensity L_o with a nominal real-valued range of 0 to 1. For interpretation of entries in Table E.4 that are expressed in terms of multiple curve segments parameterized by the variable α over a region bounded by the variable β or by the variables β and γ, the values of α and β are defined to be the positive constants necessary for the curve segments that meet at the value β to have continuity of value and continuity of slope at the value β, and the value of γ, when applicable, is defined to be the positive constant necessary for the associated curve segments to meet at the value γ. For example, for transfer_characteristics equal to 1, 6, 11, 14, or 15, α has the value $1 + 5.5 * \beta = 1.099\ 296\ 826\ 809\ 442...$ and β has the value 0.018 053 968 510 807....

When the transfer_characteristics syntax element is not present, the value of transfer_characteristics is inferred to be equal to 2 (the transfer characteristics are unspecified or are determined by the application). Values of transfer_characteristics that are identified as reserved in Table E.4 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of transfer_characteristics as equivalent to the value 2.

NOTE 5 – As indicated in Table E.4, some values of transfer_characteristics are defined in terms of a reference opto-electronic transfer characteristic function and others are defined in terms of a reference electro-optical transfer characteristic function, according to the convention that has been applied in other Specifications. In the cases of Rec. ITU-R BT.709-6 and Rec. ITU-R BT.2020-2 (which may be indicated by transfer_characteristics equal to 1, 6, 14, or 15), although the value is defined in terms of a reference opto-electronic transfer characteristic function, a suggested corresponding reference electro-optical transfer characteristic function for flat panel displays used in HDTV studio production has been specified in Rec. ITU-R BT.1886-0.

Table E.4 – Transfer characteristics interpretation using the transfer_characteristics syntax element

Value	Transfer characteristic		Informative remark
0	Reserved		For future use by ITU-T | ISO/IEC
1	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ $V = 4.500 * L_c$	for $1 \ >= \ L_c \ >= \ \beta$ for $\beta > L_c \ >= \ 0$	Rec. ITU-R BT.709-6 Rec. ITU-R BT.1361-0 conventional colour gamut system (historical) (functionally the same as the values 6, 14, and 15)
2	Unspecified		Image characteristics are unknown or are determined by the application.
3	Reserved		For future use by ITU-T | ISO/IEC
4	Assumed display gamma 2.2		Rec. ITU-R BT.470-6 System M (historical) NTSC Recommendation for transmission standards for colour television (1953) FCC, Title 47 Code of Federal Regulations (2003) 73.682 (a) (20)

Table E.4 – Transfer characteristics interpretation using the transfer_characteristics syntax element

Value	Transfer characteristic		Informative remark
5	Assumed display gamma 2.8		Rec. ITU-R BT.470-6 System B, G (historical) Rec. ITU-R BT.1700-0 625 PAL and 625 SECAM
6	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ $V = 4.500 * L_c$	for $1 >= L_c >= \beta$ for $\beta > L_c >= 0$	Rec. ITU-R BT.601-7 525 or 625 Rec. ITU-R BT.1358-1 525 or 625 (historical) Rec. ITU-R BT.1700-0 NTSC SMPTE ST 170 (2004) (functionally the same as the values 1, 14, and 15)
7	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ $V = 4.0 * L_c$	for $1 >= L_c >= \beta$ for $\beta > L_c >= 0$	SMPTE ST 240 (1999, historical)
8	$V = L_c$	for all values of L_c	Linear transfer characteristics
9	$V = 1.0 + \mathrm{Log10}(L_c) \div 2$ $V = 0.0$	for $1 >= L_c >= 0.01$ for $0.01 > L_c >= 0$	Logarithmic transfer characteristic (100:1 range)
10	$V = 1.0 + \mathrm{Log10}(L_c) \div 2.5$ $V = 0.0$	for $1 >= L_c >= \mathrm{Sqrt}(10) \div 1\,000$ for $\mathrm{Sqrt}(10) \div 1\,000 > L_c >= 0$	Logarithmic transfer characteristic (100 * Sqrt(10) : 1 range)
11	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ $V = 4.500 * L_c$ $V = -\alpha * (-L_c)^{0.45} + (\alpha - 1)$	for $L_c >= \beta$ for $\beta > L_c > -\beta$ for $-\beta >= L_c$	IEC 61966-2-4
12	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ $V = 4.500 * L_c$ $V = -(\alpha * (-4 * L_c)^{0.45} - (\alpha - 1)) \div 4$	for $1.33 > L_c >= \beta$ for $\beta > L_c >= -\gamma$ for $-\gamma > L_c >= -0.25$	Rec. ITU-R BT.1361-0 extended colour gamut system (historical)
13	– If matrix_coeffs is equal to 0 $V = \alpha * L_c^{(1 \div 2.4)} - (\alpha - 1)$ $V = 12.92 * L_c$ – Otherwise $V = \alpha * L_c^{(1 \div 2.4)} - (\alpha - 1)$ $V = 12.92 * L_c$ $V = - \alpha * (-L_c)^{(1 \div 2.4)} + (\alpha - 1)$	 for $1 >= L_c >= \beta$ for $\beta > L_c >= 0$ for $L_c >= \beta$ for $\beta > L_c > -\beta$ for $-\beta >= L_c$	IEC 61966-2-1 sRGB (with matrix_coeffs equal to 0) IEC 61966-2-1 sYCC (with matrix_coeffs equal to 5)
14	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ $V = 4.500 * L_c$	for $1 >= L_c >= \beta$ for $\beta > L_c >= 0$	Rec. ITU-R BT.2020-2 (functionally the same as the values 1, 6, and 15)
15	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ $V = 4.500 * L_c$	for $1 >= L_c >= \beta$ for $\beta > L_c >= 0$	Rec. ITU-R BT.2020-2 (functionally the same as the values 1, 6, and 14)
16	$V = ((c_1 + c_2 * L_o^n) \div (1 + c_3 * L_o^n))^m$ $c_1 = c_3 - c_2 + 1 = 3\,424 \div 4\,096 = 0.835\,937\,5$ $c_2 = 32 * 2\,413 \div 4\,096 = 18.851\,562\,5$ $c_3 = 32 * 2\,392 \div 4\,096 = 18.687\,5$ $m = 128 * 2\,523 \div 4\,096 = 78.843\,75$ $n = 0.25 * 2\,610 \div 4\,096 = 0.159\,301\,757\,812\,5$ for which L_o equal to 1 for peak white is ordinarily intended to correspond to a reference output luminance level of 10 000 candelas per square metre	for all values of L_o	SMPTE ST 2084 (2014) for 10, 12, 14, and 16-bit systems Rec. ITU-R BT.2100-2 perceptual quantization (PQ) system
17	$V = (48 * L_o \div 52.37)^{(1 \div 2.6)}$ for which L_o equal to 1 for peak white is ordinarily intended to correspond to a reference output luminance level of 48 candelas per square metre	for all values of L_o	SMPTE ST 428-1 (2006)
18	$V = a * \mathrm{Ln}(12 * L_c - b) + c$ $V = \mathrm{Sqrt}(3) * L_c^{0.5}$ $a = 0.178\,832\,77$, $b = 0.284\,668\,92$, $c = 0.559\,910\,73$	for $1 >= L_c > 1 \div 12$ for $1 \div 12 >= L_c >= 0$	Association of Radio Industries and Businesses (ARIB) STD-B67 Rec. ITU-R BT.2100-2 hybrid log-gamma (HLG) system
19..255	Reserved		For future use by ITU-T | ISO/IEC

NOTE 6 – For transfer_characteristics equal to 18, the equations given in Table E.4 are normalized for a source input linear optical intensity L_c with a nominal real-valued range of 0 to 1. An alternative scaling that is mathematically equivalent is used in ARIB STD-B67 with the source input linear optical intensity having a nominal real-valued range of 0 to 12.

matrix_coeffs describes the matrix coefficients used in deriving luma and chroma signals from the green, blue, and red, or Y, Z, and X primaries, as specified in Table E.5.

matrix_coeffs shall not be equal to 0 unless both of the following conditions are true:

– BitDepth_C is equal to BitDepth_Y.

– chroma_format_idc is equal to 3 (the 4:4:4 chroma format).

The specification of the use of matrix_coeffs equal to 0 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

matrix_coeffs shall not be equal to 8 unless one of the following conditions is true:

– BitDepth_C is equal to BitDepth_Y,

– BitDepth_C is equal to $\text{BitDepth}_Y + 1$ and chroma_format_idc is equal to 3 (the 4:4:4 chroma format).

The specification of the use of matrix_coeffs equal to 8 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

When the matrix_coeffs syntax element is not present, the value of matrix_coeffs is inferred to be equal to 2 (unspecified).

The interpretation of matrix_coeffs, together with colour_primaries and transfer_characteristics, is specified by the equations below.

NOTE 7 – For purposes of YZX representation when matrix_coeffs is equal to 0, the symbols R, G, and B are substituted for X, Y, and Z, respectively, in the following descriptions of Equations E-1 to E-3, E-13 to E-15, E-19 to E-21, and E-31 to E-33.

E_R, E_G, and E_B are defined as "linear-domain" real-valued signals based on the indicated colour primaries before application of the transfer characteristics function.

Nominal peak white is specified as having E_R equal to 1, E_G equal to 1, and E_B equal to 1.

Nominal black is specified as having E_R equal to 0, E_G equal to 0, and E_B equal to 0.

The application of the transfer characteristics function is denoted by $(x)'$ for an argument x.

– If matrix_coeffs is not equal to 14, the signals E'_R, E'_G, and E'_B are determined by application of the transfer characteristics function as follows:

$$E'_R = (E_R)' \quad \text{(E-1)}$$

$$E'_G = (E_G)' \quad \text{(E-2)}$$

$$E'_B = (E_B)' \quad \text{(E-3)}$$

In this case, the range of E'_R, E'_G, and E'_B is specified as follows:

  – If transfer_characteristics is equal to 11 or 12, or transfer_characteristics is equal to 13 and matrix_coeffs is not equal to 0, E'_R, E'_G, and E'_B are real numbers with values that have a larger range than the range of 0 to 1, inclusive, and their range is not specified in this Specification.

  – Otherwise, E'_R, E'_G and E'_B are real numbers in the range of 0 to 1.

– Otherwise (matrix_coeffs is equal to 14), the "linear-domain" real-valued signals E_L, E_M, and E_S are determined as follows:

$$E_L = (1\,688 * E_R + 2\,146 * E_G + 262 * E_B) \div 4\,096 \quad \text{(E-4)}$$

$$E_M = (683 * E_R + 2\,951 * E_G + 462 * E_B) \div 4\,096 \quad \text{(E-5)}$$

$$E_S = (99 * E_R + 309 * E_G + 3\,688 * E_B) \div 4\,096 \quad \text{(E-6)}$$

In this case, the signals E'_L, E'_M, and E'_S are determined by application of the transfer characteristics function as follows:

$$E'_L = (E_L)' \quad (E-7)$$

$$E'_M = (E_M)' \quad (E-8)$$

$$E'_S = (E_S)' \quad (E-9)$$

The interpretation of matrix_coeffs is specified as follows:

– If video_full_range_flag is equal to 0, the following applies:
  – If matrix_coeffs is equal to 1, 4, 5, 6, 7, 9, 10, 11, 12, 13, or 14, the following equations apply:

$$Y = \mathrm{Clip1}_Y(\mathrm{Round}((1 \ll (\mathrm{BitDepth}_Y - 8)) * (219 * E'_Y + 16))) \quad (E-10)$$

$$Cb = \mathrm{Clip1}_C(\mathrm{Round}((1 \ll (\mathrm{BitDepth}_C - 8)) * (224 * E'_{PB} + 128))) \quad (E-11)$$

$$Cr = \mathrm{Clip1}_C(\mathrm{Round}((1 \ll (\mathrm{BitDepth}_C - 8)) * (224 * E'_{PR} + 128))) \quad (E-12)$$

  – Otherwise, if matrix_coeffs is equal to 0 or 8, the following equations apply:

$$R = \mathrm{Clip1}_Y((1 \ll (\mathrm{BitDepth}_Y - 8)) * (219 * E'_R + 16)) \quad (E-13)$$

$$G = \mathrm{Clip1}_Y((1 \ll (\mathrm{BitDepth}_Y - 8)) * (219 * E'_G + 16)) \quad (E-14)$$

$$B = \mathrm{Clip1}_Y((1 \ll (\mathrm{BitDepth}_Y - 8)) * (219 * E'_B + 16)) \quad (E-15)$$

  – Otherwise, if matrix_coeffs is equal to 2, the interpretation of the matrix_coeffs syntax element is unknown or is determined by the application.
  – Otherwise (matrix_coeffs is not equal to 0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, or 14), the interpretation of the matrix_coeffs syntax element is reserved for future definition by ITU-T | ISO/IEC.

– Otherwise (video_full_range_flag is equal to 1), the following applies:
  – If matrix_coeffs is equal to 1, 4, 5, 6, 7, 9, 10, 11, 12, 13, or 14, the following applies:

$$Y = \mathrm{Clip1}_Y(\mathrm{Round}(((1 \ll \mathrm{BitDepth}_Y) - 1) * E'_Y)) \quad (E-16)$$

$$Cb = \mathrm{Clip1}_C(\mathrm{Round}(((1 \ll \mathrm{BitDepth}_C) - 1) * E'_{PB} + (1 \ll (\mathrm{BitDepth}_C - 1)))) \quad (E-17)$$

$$Cr = \mathrm{Clip1}_C(\mathrm{Round}(((1 \ll \mathrm{BitDepth}_C) - 1) * E'_{PR} + (1 \ll (\mathrm{BitDepth}_C - 1)))) \quad (E-18)$$

  – Otherwise, if matrix_coeffs is equal to 0 or 8, the following applies:

$$R = \mathrm{Clip1}_Y(((1 \ll \mathrm{BitDepth}_Y) - 1) * E'_R) \quad (E-19)$$

$$G = \mathrm{Clip1}_Y(((1 \ll \mathrm{BitDepth}_Y) - 1) * E'_G) \quad (E-20)$$

$$B = \mathrm{Clip1}_Y(((1 \ll \mathrm{BitDepth}_Y) - 1) * E'_B) \quad (E-21)$$

  – Otherwise, if matrix_coeffs is equal to 2, the interpretation of the matrix_coeffs syntax element is unknown or is determined by the application.
  – Otherwise (matrix_coeffs is not equal to 0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, or 14), the interpretation of the matrix_coeffs syntax element is reserved for future definition by ITU-T | ISO/IEC. Reserved values for matrix_coeffs shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of matrix_coeffs as equivalent to the value 2.

It is a requirement of bitstream conformance to this version of this Specification that when colour_primaries is not equal to 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, or 22, matrix_coeffs shall not be equal to 12 or 13.

When matrix_coeffs is equal to 1, 4, 5, 6, 7, 9, 10, 11, 12, or 13, the constants K_B and K_R are specified as follows:

– If matrix_coeffs is not equal to 12 or 13, the constants K_B and K_R are specified in Table E.5.

– Otherwise (matrix_coeffs is equal to 12 or 13), the constants K_R and K_B are computed as follows, using the chromaticity coordinates (x_R, y_R), (x_G, y_G), (x_B, y_B), and (x_W, y_W) specified by Table E.3 for the colour_primaries syntax element for the red, green, blue, and white colour primaries, respectively.

$$K_R = \frac{y_R * (x_W * (y_G * z_B - y_B * z_G) + y_W * (x_B * z_G - x_G * z_B) + z_W * (x_G * y_B - x_B * y_G))}{y_W * (x_R * (y_G * z_B - y_B * z_G) + x_G * (y_B * z_R - y_R * z_B) + x_B * (y_R * z_G - y_G * z_R))} \quad \text{(E-22)}$$

$$K_B = \frac{y_B * (x_W * (y_R * z_G - y_G * z_R) + y_W * (x_G * z_R - x_R * z_G) + z_W * (x_R * y_G - x_G * y_R))}{y_W * (x_R * (y_G * z_B - y_B * z_G) + x_G * (y_B * z_R - y_R * z_B) + x_B * (y_R * z_G - y_G * z_R))} \quad \text{(E-23)}$$

where the values of z_R, z_G, z_B, and z_W, are given by.

$$z_R = 1 - (x_R + y_R) \quad \text{(E-24)}$$

$$z_G = 1 - (x_G + y_G) \quad \text{(E-25)}$$

$$z_B = 1 - (x_B + y_B) \quad \text{(E-26)}$$

$$z_W = 1 - (x_W + y_W) \quad \text{(E-27)}$$

The variables E'_Y, E'_{PB}, and E'_{PR} (for matrix_coeffs not equal to 0 or 8) or Y, Cb, and Cr (for matrix_coeffs equal to 0 or 8) are specified as follows:

– If matrix_coeffs is not equal to 0, 8, 10, 11, 13, or 14, the following equations apply:

$$E'_Y = K_R * E'_R + (1 - K_R - K_B) * E'_G + K_B * E'_B \quad \text{(E-28)}$$

$$E'_{PB} = 0.5 * (E'_B - E'_Y) \div (1 - K_B) \quad \text{(E-29)}$$

$$E'_{PR} = 0.5 * (E'_R - E'_Y) \div (1 - K_R) \quad \text{(E-30)}$$

NOTE 8 – E'_Y is a real number with the value 0 associated with nominal black and the value 1 associated with nominal white. E'_{PB} and E'_{PR} are real numbers with the value 0 associated with both nominal black and nominal white. When transfer_characteristics is not equal to 11 or 12, E'_Y is a real number with values in the range of 0 to 1 inclusive. When transfer_characteristics is not equal to 11 or 12, E'_{PB} and E'_{PR} are real numbers with values in the range of −0.5 to 0.5 inclusive. When transfer_characteristics is equal to 11 or 12, E'_Y, E'_{PB}, and E'_{PR} are real numbers with a larger range not specified in this Specification.

– Otherwise, if matrix_coeffs is equal to 0, the following equations apply:

$$Y = \text{Round}(G) \quad \text{(E-31)}$$

$$Cb = \text{Round}(B) \quad \text{(E-32)}$$

$$Cr = \text{Round}(R) \quad \text{(E-33)}$$

– Otherwise, if matrix_coeffs is equal to 8, the following applies:

 – If $BitDepth_C$ is equal to $BitDepth_Y$, the following equations apply:

$$Y = \text{Round}(0.5 * G + 0.25 * (R + B)) \quad \text{(E-34)}$$

$$Cb = \text{Round}(0.5 * G - 0.25 * (R + B)) + (1 \ll (BitDepth_C - 1)) \quad \text{(E-35)}$$

$$Cr = \text{Round}(0.5 * (R - B)) + (1 \ll (BitDepth_C - 1)) \quad \text{(E-36)}$$

NOTE 9 – In this case, for purposes of the YCgCo nomenclature used in Table E.5, Cb and Cr of Equations E-35 and E-36 may be referred to as Cg and Co, respectively. An appropriate inverse conversion for Equations E-34 to E-36 is as follows:

$$t = Y - (Cb - (1 \ll (BitDepth_C - 1))) \quad \text{(E-37)}$$

$$G = \text{Clip1}_Y(Y + (Cb - (1 \ll (BitDepth_C - 1)))) \quad \text{(E-38)}$$

$$B = \text{Clip1}_Y(t - (Cr - (1 \ll (BitDepth_C - 1)))) \quad \text{(E-39)}$$

$$R = \text{Clip1}_Y(t + (Cr - (1 \ll (BitDepth_C - 1)))) \quad \text{(E-40)}$$

– Otherwise ($BitDepth_C$ is not equal to $BitDepth_Y$), the following equations apply:

$$Cr = Round(R) - Round(B) + (1 \ll (BitDepth_C - 1)) \tag{E-41}$$

$$t = Round(B) + ((Cr - (1 \ll (BitDepth_C - 1))) \gg 1) \tag{E-42}$$

$$Cb = Round(G) - t + (1 \ll (BitDepth_C - 1)) \tag{E-43}$$

$$Y = t + ((Cb - (1 \ll (BitDepth_C - 1))) \gg 1) \tag{E-44}$$

NOTE 10 – In this case, for purposes of the YCgCo nomenclature used in Table E.5, Cb and Cr of Equations E-43 and E-41 may be referred to as Cg and Co, respectively. An appropriate inverse conversion for Equations E-41 to E-44 is as follows:

$$t = Y - ((Cb - (1 \ll (BitDepth_C - 1))) \gg 1) \tag{E-45}$$

$$G = Clip1_Y(t + (Cb - (1 \ll (BitDepth_C - 1)))) \tag{E-46}$$

$$B = Clip1_Y(t - ((Cr - (1 \ll (BitDepth_C - 1))) \gg 1)) \tag{E-47}$$

$$R = Clip1_Y(B + (Cr - (1 \ll (BitDepth_C - 1)))) \tag{E-48}$$

– Otherwise, if matrix_coeffs is equal to 10 or 13, the signal E'_Y is determined by application of the transfer characteristics function as follows and Equations E-51 to E-54 apply for specification of the signals E'_{PB} and E'_{PR}:

$$E_Y = K_R * E_R + (1 - K_R - K_B) * E_G + K_B * E_B \tag{E-49}$$

$$E'_Y = (E_Y)' \tag{E-50}$$

NOTE 11 – In this case, E_Y is defined from the "linear-domain" signals for E_R, E_G, and E_B, prior to application of the transfer characteristics function, which is then applied to produce the signal E'_Y. E_Y and E'_Y are real values with the value 0 associated with nominal black and the value 1 associated with nominal white.

while the signals E'_{PB} and E'_{PR} are determined as follows:

$$E'_{PB} = (E'_B - E'_Y) \div (2 * N_B) \quad \text{for } -N_B \le E'_B - E'_Y \le 0 \tag{E-51}$$

$$E'_{PB} = (E'_B - E'_Y) \div (2 * P_B) \quad \text{for } 0 < E'_B - E'_Y \le P_B \tag{E-52}$$

$$E'_{PR} = (E'_R - E'_Y) \div (2 * N_R) \quad \text{for } -N_R \le E'_R - E'_Y \le 0 \tag{E-53}$$

$$E'_{PR} = (E'_R - E'_Y) \div (2 * P_R) \quad \text{for } 0 < E'_R - E'_Y \le P_R \tag{E-54}$$

where the constants N_B, P_B, N_R, and P_R are determined by application of the transfer characteristics function to expressions involving the constants K_B and K_R as follows:

$$N_B = (1 - K_B)' \tag{E-55}$$

$$P_B = 1 - (K_B)' \tag{E-56}$$

$$N_R = (1 - K_R)' \tag{E-57}$$

$$P_R = 1 - (K_R)' \tag{E-58}$$

– Otherwise, if matrix_coeffs is equal to 11, the following equations apply:

$$E'_Y = E'_G \tag{E-59}$$

$$E'_{PB} = 0.5 * (0.986\,566 * E'_B - E'_Y) \tag{E-60}$$

$$E'_{PR} = 0.5 * (E'_{R} - 0.991\,902 * E'_{Y}) \tag{E-61}$$

NOTE 12 – In this case, for purposes of the $Y'D'_{Z}D'_{X}$ nomenclature used in Table E.5, E'_{PB} may be referred to as D'_{Z} and E'_{PR} may be referred to as D'_{X}.

– Otherwise (matrix_coeffs is equal to 14), the following equations apply:

– If transfer_characteristics is not equal to 18, the following equations apply:

$$E'_{Y} = 0.5 * (E'_{L} + E'_{M}) \tag{E-62}$$

$$E'_{PB} = (6\,610 * E'_{L} - 13\,613 * E'_{M} + 7\,003 * E'_{S}) \div 4\,096 \tag{E-63}$$

$$E'_{PR} = (17\,933 * E'_{L} - 17\,390 * E'_{M} - 543 * E'_{S}) \div 4\,096 \tag{E-64}$$

– Otherwise, the following equations apply:

$$E'_{Y} = 0.5 * (E'_{L} + E'_{M}) \tag{E-65}$$

$$E'_{PB} = (3\,625 * E'_{L} - 7\,465 * E'_{M} + 3\,840 * E'_{S}) \div 4\,096 \tag{E-66}$$

$$E'_{PR} = (9\,500 * E'_{L} - 9\,212 * E'_{M} - 288 * E'_{S}) \div 4\,096 \tag{E-67}$$

NOTE 13 – In this case, for purposes of the $IC_{T}C_{P}$ nomenclature used in Table E.5, E'_{Y}, E'_{PB}, and E'_{PR} of Equations E-62, E-63, and E-64 or Equations E-65, E-66, and E-67 may be referred to as I, C_{T}, and C_{P}, respectively. Equations E-62 to E-64 were designed specifically for use with transfer_characteristics equal to 16 (PQ), and Equations E-65 to E-67 were designed specifically for use with transfer_characteristics equal to 18 (HLG).

Table E.5 – Matrix coefficients interpretation using the matrix_coeffs syntax element

Value	Matrix	Informative remark
0	Identity	The identity matrix. Typically used for GBR (often referred to as RGB); however, may also be used for YZX (often referred to as XYZ) IEC 61966-2-1 sRGB SMPTE ST 428-1 (2006) See Equations E-31 to E-33
1	$K_R = 0.2126$; $K_B = 0.0722$	Rec. ITU-R BT.709-6 Rec. ITU-R BT.1361-0 conventional colour gamut system and extended colour gamut system (historical) IEC 61966-2-4 $xvYCC_{709}$ SMPTE RP 177 (1993) Annex B See Equations E-28 to E-30
2	Unspecified	Image characteristics are unknown or are determined by the application.
3	Reserved	For future use by ITU-T | ISO/IEC
4	$K_R = 0.30$; $K_B = 0.11$	FCC Title 47 Code of Federal Regulations (2003) 73.682 (a) (20) See Equations E-28 to E-30
5	$K_R = 0.299$; $K_B = 0.114$	Rec. ITU-R BT.470-6 System B, G (historical) Rec. ITU-R BT.601-7 625 Rec. ITU-R BT.1358-0 625 (historical) Rec. ITU-R BT.1700-0 625 PAL and 625 SECAM IEC 61966-2-1 sYCC IEC 61966-2-4 $xvYCC_{601}$ (functionally the same as the value 6) See Equations E-28 to E-30
6	$K_R = 0.299$; $K_B = 0.114$	Rec. ITU-R BT.601-7 525 Rec. ITU-R BT.1358-1 525 or 625 (historical) Rec. ITU-R BT.1700-0 NTSC SMPTE ST 170 (2004) (functionally the same as the value 5) See Equations E-28 to E-30
7	$K_R = 0.212$; $K_B = 0.087$	SMPTE ST 240 (1999, historical) See Equations E-28 to E-30
8	YCgCo	See Equations E-34 to E-48
9	$K_R = 0.2627$; $K_B = 0.0593$	Rec. ITU-R BT.2020-2 non-constant luminance system Rec. ITU-R BT.2100-2 Y′CbCr See Equations E-28 to E-30
10	$K_R = 0.2627$; $K_B = 0.0593$	Rec. ITU-R BT.2020-2 constant luminance system See Equations E-49 to E-58
11	$Y'D'_ZD'_X$	SMPTE ST 2085 (2015) See Equations E-59 to E-61
12	See Equations E-22 to E-27	Chromaticity-derived non-constant luminance system See Equations E-28 to E-30
13	See Equations E-22 to E-27	Chromaticity-derived constant luminance system See Equations E-49 to E-58
14	IC_TC_P	Rec. ITU-R BT.2100-2 IC_TC_P See Equations E-62 to E-64 for transfer_characteristics value 16 (PQ) See Equations E-65 to E-67 for transfer_characteristics value 18 (HLG)
15..255	Reserved	For future use by ITU-T | ISO/IEC

chroma_loc_info_present_flag equal to 1 specifies that chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field are present. chroma_loc_info_present_flag equal to 0 specifies that chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field are not present.

When chroma_format_idc is not equal to 1, chroma_loc_info_present_flag should be equal to 0.

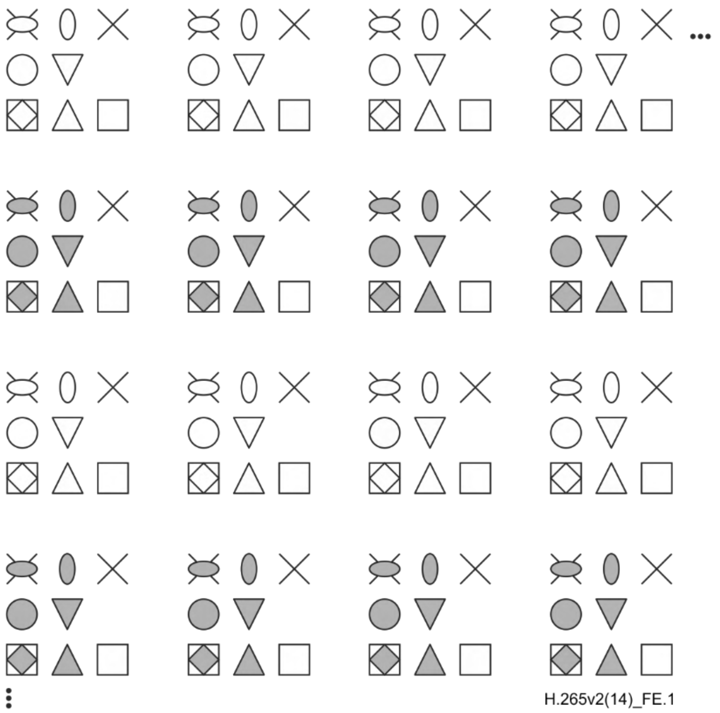
chroma_sample_loc_type_top_field and **chroma_sample_loc_type_bottom_field** specify the location of chroma samples as follows:

– If chroma_format_idc is equal to 1 (4:2:0 chroma format), chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field specify the location of chroma samples for the top field and the bottom field, respectively, as shown in Figure E.1.

– Otherwise (chroma_format_idc is not equal to 1), the values of the syntax elements chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field shall be ignored. When chroma_format_idc is equal to 2 (4:2:2 chroma format) or 3 (4:4:4 chroma format), the location of chroma samples is specified in clause 6.2. When chroma_format_idc is equal to 0, there is no chroma sample array.

The value of chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field shall be in the range of 0 to 5, inclusive. When the chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field are not present, the values of chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field are inferred to be equal to 0.

NOTE 14 – When coding progressive source material, chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field should have the same value.



Interpretation of symbols

Luma sample position indications:

Luma sample top field

Luma sample bottom field

Chroma sample position indications, where gray fill indicates a bottom field sample type and no fill indicates a top field sample type:

Chroma sample type 2

Chroma sample type 3

Chroma sample type 0

Chroma sample type 1

Chroma sample type 4

Chroma sample type 5

Figure E.1 – Location of chroma samples for top and bottom fields for chroma_format_idc equal to 1 (4:2:0 chroma format) as a function of chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field

Figure E.2 illustrates the indicated relative position of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format), and chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field are both

equal to the value of a variable ChromaLocType. The region represented by the top-left 4:2:0 chroma sample (depicted as a large red square with a large red dot at its centre) is shown relative to the region represented by the top-left luma sample (depicted as a small black square with a small black dot at its centre). The regions represented by neighbouring luma samples are depicted as small grey squares with small grey dots at their centres.

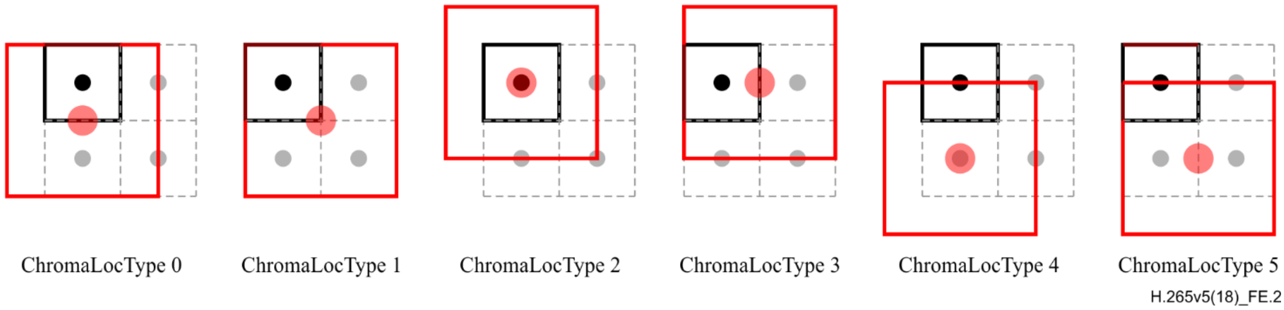


Figure E.2 – Location of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format) as a function of ChromaLocType

The relative spatial positioning of the chroma samples, as illustrated in Figure E.3, can be expressed by defining two variables HorizontalOffsetC and VerticalOffsetC as a function of chroma_format_idc and the variable ChromaLocType as given by Table E.6, where HorizontalOffsetC is the horizontal (x) position of the centre of the top-left chroma sample relative to the centre of the top-left luma sample in units of luma samples and VerticalOffsetC is the vertical (y) position of the centre of the top-left chroma sample relative to the centre of the top-left luma sample in units of luma samples.

In a typical FIR filter design, when chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), HorizontalOffsetC and VerticalOffsetC would serve as the phase offsets for the horizontal and vertical filter operations, respectively, for separable downsampling from 4:4:4 chroma format to the chroma format indicated by chroma_format_idc.

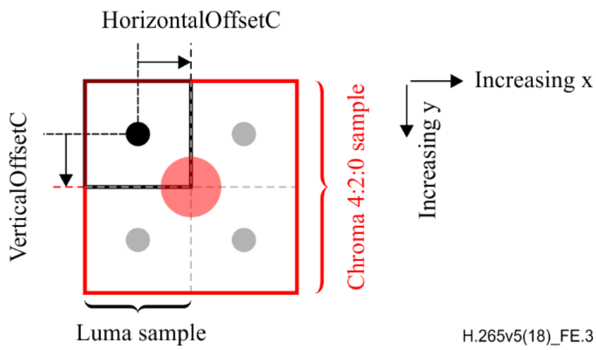


Figure E.3 – Location of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format) when ChromaLocType is equal to 1

Table E.6 – Definition of HorizontalOffsetC and VerticalOffsetC as a function of chroma_format_idc and ChromaLocType

chroma_format_idc	ChromaLocType	HorizontalOffsetC	VerticalOffsetC
1 (4:2:0)	0	0	0.5
1 (4:2:0)	1	0.5	0.5
1 (4:2:0)	2	0	0
1 (4:2:0)	3	0.5	0
1 (4:2:0)	4	0	1
1 (4:2:0)	5	0.5	1
2 (4:2:2)	–	0	0
3 (4:4:4)	–	0	0

When chroma_format_idc is equal to 1 (4:2:0 chroma format) and the decoded video content is intended for interpretation according to Rec. ITU-R BT.2020-2 or Rec. ITU-R BT.2100-2, chroma_loc_info_present_flag should be equal to 1, and chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field should both be equal to 2.

neutral_chroma_indication_flag equal to 1 indicates that the value of all decoded chroma samples is equal to $1 << (\text{BitDepth}_C - 1)$. neutral_chroma_indication_flag equal to 0 provides no indication of decoded chroma sample values. When neutral_chroma_indication_flag is equal to 1, it is a requirement of bitstream conformance that the value of all decoded chroma samples produced by the decoding process shall be equal to $1 << (\text{BitDepth}_C - 1)$. When neutral_chroma_indication_flag is not present, it is inferred to be equal to 0.

NOTE 15 – When neutral_chroma_indication_flag is equal to 1, it is not necessary for the decoder to apply the specified decoding process in order to determine the value of the decoded chroma samples.

field_seq_flag equal to 1 indicates that the CVS conveys pictures that represent fields, and specifies that a picture timing SEI message shall be present in every access unit of the current CVS. field_seq_flag equal to 0 indicates that the CVS conveys pictures that represent frames and that a picture timing SEI message may or may not be present in any access unit of the current CVS. When field_seq_flag is not present, it is inferred to be equal to 0. When general_frame_only_constraint_flag is equal to 1, the value of field_seq_flag shall be equal to 0.

NOTE 16 – The specified decoding process does not treat access units conveying pictures that represent fields or frames differently. A sequence of pictures that represent fields would therefore be coded with the picture dimensions of an individual field. For example, access units containing pictures that represent 1080i fields would commonly have cropped output dimensions of 1 920x540, while the sequence picture rate would commonly express the rate of the source fields (typically between 50 and 60 Hz), instead of the source frame rate (typically between 25 and 30 Hz).

frame_field_info_present_flag equal to 1 specifies that picture timing SEI messages are present for every picture and include the pic_struct, source_scan_type and duplicate_flag syntax elements. frame_field_info_present_flag equal to 0 specifies that the pic_struct syntax element is not present in picture timing SEI messages.

When frame_field_info_present_flag is present and either or both of the following conditions are true, frame_field_info_present_flag shall be equal to 1:

– field_seq_flag is equal to 1.

– general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 1.

When frame_field_info_present_flag is not present, its value is inferred as follows:

– If general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 1, frame_field_info_present_flag is inferred to be equal to 1.

– Otherwise, frame_field_info_present_flag is inferred to be equal to 0.

default_display_window_flag equal to 1 indicates that the default display window parameters follow next in the VUI. default_display_window_flag equal to 0 indicates that the default display window parameters are not present. The default display window parameters identify the area that is within the conformance cropping window and that is suggested to be displayed in the absence of any alternative indication (provided within the bitstream or by external means not specified in this Specification) of preferred display characteristics.

def_disp_win_left_offset, **def_disp_win_right_offset**, **def_disp_win_top_offset** and **def_disp_win_bottom_offset** specify the samples of the pictures in the CVS that are within the default display window, in terms of a rectangular region specified in picture coordinates for display. When default_display_window_flag is equal to 0, the values of def_disp_win_left_offset, def_disp_win_right_offset, def_disp_win_top_offset and def_disp_win_bottom_offset are inferred to be equal to 0.

The following variables are derived from the default display window parameters:

$$\text{leftOffset} = \text{conf\_win\_left\_offset} + \text{def\_disp\_win\_left\_offset} \quad \text{(E-68)}$$

$$\text{rightOffset} = \text{conf\_win\_right\_offset} + \text{def\_disp\_win\_right\_offset} \quad \text{(E-69)}$$

$$\text{topOffset} = \text{conf\_win\_top\_offset} + \text{def\_disp\_win\_top\_offset} \quad \text{(E-70)}$$

$$\text{bottomOffset} = \text{conf\_win\_bottom\_offset} + \text{def\_disp\_win\_bottom\_offset} \quad \text{(E-71)}$$

The default display window contains the luma samples with horizontal picture coordinates from SubWidthC * leftOffset to pic_width_in_luma_samples − (SubWidthC * rightOffset + 1) and vertical picture coordinates from SubHeightC * topOffset to pic_height_in_luma_samples − (SubHeightC * bottomOffset + 1), inclusive.

The value of SubWidthC * (leftOffset + rightOffset) shall be less than pic_width_in_luma_samples and the value of SubHeightC * (topOffset + bottomOffset) shall be less than pic_height_in_luma_samples.

When ChromaArrayType is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates (x / SubWidthC, y / SubHeightC), where (x, y) are the picture coordinates of the specified luma samples.

vui_timing_info_present_flag equal to 1 specifies that vui_num_units_in_tick, vui_time_scale, vui_poc_proportional_to_timing_flag and vui_hrd_parameters_present_flag are present in the vui_parameters( ) syntax structure. vui_timing_info_present_flag equal to 0 specifies that vui_num_units_in_tick, vui_time_scale, vui_poc_proportional_to_timing_flag and vui_hrd_parameters_present_flag are not present in the vui_parameters( ) syntax structure.

vui_num_units_in_tick is the number of time units of a clock operating at the frequency vui_time_scale Hz that corresponds to one increment (called a clock tick) of a clock tick counter. vui_num_units_in_tick shall be greater than 0. A clock tick, in units of seconds, is equal to the quotient of vui_num_units_in_tick divided by vui_time_scale. For example, when the picture rate of a video signal is 25 Hz, vui_time_scale may be equal to 27 000 000 and vui_num_units_in_tick may be equal to 1 080 000 and consequently a clock tick may be equal to 0.04 seconds.

When vps_num_units_in_tick is present in the VPS referred to by the SPS, vui_num_units_in_tick, when present, shall be equal to vps_num_units_in_tick, and when not present, is inferred to be equal to vps_num_units_in_tick.

vui_time_scale is the number of time units that pass in one second. For example, a time coordinate system that measures time using a 27 MHz clock has a vui_time_scale of 27 000 000. The value of vui_time_scale shall be greater than 0.

When vps_time_scale is present in the VPS referred to by the SPS, vui_time_scale, when present, shall be equal to vps_time_scale, and when not present, is inferred to be equal to vps_time_scale.

vui_poc_proportional_to_timing_flag equal to 1 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, is proportional to the output time of the picture relative to the output time of the first picture in the CVS. vui_poc_proportional_to_timing_flag equal to 0 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, may or may not be proportional to the output time of the picture relative to the output time of the first picture in the CVS.

When vps_poc_proportional_to_timing_flag is present in the VPS referred to by the SPS and the value is equal to 1, vui_poc_proportional_to_timing_flag, when present, shall be equal to 1.

vui_num_ticks_poc_diff_one_minus1 plus 1 specifies the number of clock ticks corresponding to a difference of picture order count values equal to 1. The value of vui_num_ticks_poc_diff_one_minus1 shall be in the range of 0 to $2^{32} - 2$, inclusive.

When vps_num_ticks_poc_diff_one_minus1 is present in the VPS referred to by the SPS, vui_num_ticks_poc_diff_one_minus1, when present, shall be equal to vps_num_ticks_poc_diff_one_minus1.

vui_hrd_parameters_present_flag equal to 1 specifies that the syntax structure hrd_parameters( ) is present in the vui_parameters( ) syntax structure. vui_hrd_parameters_present_flag equal to 0 specifies that the syntax structure hrd_parameters( ) is not present in the vui_parameters( ) syntax structure.

bitstream_restriction_flag equal to 1, specifies that the bitstream restriction parameters for the CVS are present. bitstream_restriction_flag equal to 0, specifies that the bitstream restriction parameters for the CVS are not present.

tiles_fixed_structure_flag equal to 1 indicates that each PPS that is active in the CVS has the same value of the syntax elements num_tile_columns_minus1, num_tile_rows_minus1, uniform_spacing_flag, column_width_minus1[i], row_height_minus1[i] and loop_filter_across_tiles_enabled_flag, when present. tiles_fixed_structure_flag equal to 0 indicates that tiles syntax elements in different PPSs may or may not have the same value. When the tiles_fixed_structure_flag syntax element is not present, it is inferred to be equal to 0.

NOTE 17 – The signalling of tiles_fixed_structure_flag equal to 1 is a guarantee to a decoder that each picture in the CVS has the same number of tiles distributed in the same way which might be useful for workload allocation in the case of multi-threaded decoding.

motion_vectors_over_pic_boundaries_flag equal to 0 indicates that no sample outside the picture boundaries and no sample at a fractional sample position for which the sample value is derived using one or more samples outside the picture boundaries is used for inter prediction of any sample. motion_vectors_over_pic_boundaries_flag equal to 1 indicates that one or more samples outside the picture boundaries may be used in inter prediction. When the motion_vectors_over_pic_boundaries_flag syntax element is not present, motion_vectors_over_pic_boundaries_flag value is inferred to be equal to 1.

restricted_ref_pic_lists_flag equal to 1 indicates that all P and B slices (when present) that belong to the same picture have an identical reference picture list 0 and that all B slices (when present) that belong to the same picture have an identical reference picture list 1.

min_spatial_segmentation_idc, when not equal to 0, establishes a bound on the maximum possible size of distinct coded spatial segmentation regions in the pictures of the CVS. When min_spatial_segmentation_idc is not present, it is inferred to be equal to 0. The value of min_spatial_segmentation_idc shall be in the range of 0 to 4095, inclusive.

The variable minSpatialSegmentationTimes4 is derived from min_spatial_segmentation_idc as follows:

$$\text{minSpatialSegmentationTimes4} = \text{min\_spatial\_segmentation\_idc} + 4 \quad \text{(E-72)}$$

A slice is said to contain a specific luma sample when the coding block that contains the luma sample is contained in the slice. Correspondingly, a tile is said to contain a specific luma sample when the coding block that contains the luma sample is contained in the tile.

Depending on the value of min_spatial_segmentation_idc, the following applies:

– If min_spatial_segmentation_idc is equal to 0, no limit on the maximum size of spatial segments is indicated.

– Otherwise (min_spatial_segmentation_idc is not equal to 0), it is a requirement of bitstream conformance that exactly one of the following conditions shall be true:

  – In each PPS that is activated within the CVS, tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 0 and there is no slice in the CVS that contains more than (4 * PicSizeInSamplesY) / minSpatialSegmentationTimes4 luma samples.

  – In each PPS that is activated within the CVS, tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 0 and there is no tile in the CVS that contains more than (4 * PicSizeInSamplesY) / minSpatialSegmentationTimes4 luma samples.

  – In each PPS that is activated within the CVS, tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1 and the syntax elements pic_width_in_luma_samples, pic_height_in_luma_samples and the variable CtbSizeY obey the following constraint:

$$\begin{aligned}&(2 * \text{pic\_height\_in\_luma\_samples} + \text{pic\_width\_in\_luma\_samples}) * \text{CtbSizeY}\\ &\quad <= (4 * \text{PicSizeInSamplesY}) / \text{minSpatialSegmentationTimes4}\end{aligned} \quad \text{(E-73)}$$

NOTE 18 – The syntax element min_spatial_segmentation_idc can be used by a decoder to calculate the maximum number of luma samples to be processed by one processing thread, making the assumption that the decoder maximally utilizes the parallel decoding information. However, it is important to be aware that there may be some inter-dependencies between the different threads – e.g., due to entropy coding synchronization or deblocking filtering across tile or slice boundaries. To aid decoders in planning the decoding workload distribution, encoders are encouraged to set the value of min_spatial_segmentation_idc to the highest possible value for which one of the above three conditions is true. For example, for the case when tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1, encoders should set min_spatial_segmentation_idc equal to 4 * PicSizeInSamplesY / ((2 * pic_height_in_luma_samples + pic_width_in_luma_samples) * CtbSizeY) − 4.

max_bytes_per_pic_denom indicates a number of bytes not exceeded by the sum of the sizes of the VCL NAL units associated with any coded picture in the CVS.

The number of bytes that represent a picture in the NAL unit stream is specified for this purpose as the total number of bytes of VCL NAL unit data (i.e., the total of the NumBytesInNalUnit variables for the VCL NAL units) for the picture. The value of max_bytes_per_pic_denom shall be in the range of 0 to 16, inclusive.

Depending on the value of max_bytes_per_pic_denom the following applies:

– If max_bytes_per_pic_denom is equal to 0, no limits are indicated.

– Otherwise (max_bytes_per_pic_denom is not equal to 0), it is a requirement of bitstream conformance that no coded picture shall be represented in the CVS by more than the following number of bytes.

$$(\text{PicSizeInMinCbsY} * \text{RawMinCuBits}) \div (8 * \text{max\_bytes\_per\_pic\_denom}) \quad \text{(E-74)}$$

When the max_bytes_per_pic_denom syntax element is not present, the value of max_bytes_per_pic_denom is inferred to be equal to 2.

max_bits_per_min_cu_denom indicates an upper bound for the number of coded bits of coding_unit( ) data for any coding block in any picture of the CVS. The value of max_bits_per_min_cu_denom shall be in the range of 0 to 16, inclusive.

Depending on the value of max_bits_per_min_cu_denom, the following applies:

– If max_bits_per_min_cu_denom is equal to 0, no limit is specified by this syntax element.

– Otherwise (max_bits_per_min_cu_denom is not equal to 0), it is a requirement of bitstream conformance that no coded coding_unit( ) shall be represented in the bitstream by more than the following number of bits:

$$(128 + \text{RawMinCuBits}) \div \text{max\_bits\_per\_min\_cu\_denom} * (1 \ll (2 * (\text{log2CbSize} - \text{MinCbLog2SizeY}))) \quad \text{(E-75)}$$

where log2CbSize is the value of log2CbSize for the given coding block and the number of bits of coding_unit( ) data for the same coding block is given by the number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4.

When the max_bits_per_min_cu_denom is not present, the value of max_bits_per_min_cu_denom is inferred to be equal to 1.

log2_max_mv_length_horizontal and **log2_max_mv_length_vertical** indicate the maximum absolute value of a decoded horizontal and vertical motion vector component, respectively, in quarter luma sample units, for all pictures in the CVS. A value of n asserts that no value of a motion vector component is outside the range of -2^n to $2^n - 1$, inclusive, in units of quarter luma sample displacement, where n refers to the value of log2_max_mv_length_horizontal and log2_max_mv_length_vertical for the horizontal and vertical component of the MV, respectively. The values of log2_max_mv_length_horizontal and log2_max_mv_length_vertical shall be in the range of 0 to 15, inclusive. When not present, the values of log2_max_mv_length_horizontal and log2_max_mv_length_vertical are both inferred to be equal to 15.

E.3.2 HRD parameters semantics

The hrd_parameters( ) syntax structure provides HRD parameters used in the HRD operations for a layer set. When the hrd_parameters( ) syntax structure is included in a VPS, the applicable layer set to which the hrd_parameters( ) syntax structure applies is specified by the corresponding hrd_layer_set_idx[i] syntax element in the VPS. When the hrd_parameters( ) syntax structure is included in an SPS, the layer set to which the hrd_parameters( ) syntax structure applies is the layer set for which the associated layer identifier list contains all nuh_layer_id values present in the CVS.

For interpretation of the following semantics, the bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with the layer set to which the hrd_parameters( ) syntax structure applies.

nal_hrd_parameters_present_flag equal to 1 specifies that NAL HRD parameters (pertaining to the Type II bitstream conformance point) are present in the hrd_parameters( ) syntax structure. nal_hrd_parameters_present_flag equal to 0 specifies that NAL HRD parameters are not present in the hrd_parameters( ) syntax structure.

NOTE 1 – When nal_hrd_parameters_present_flag is equal to 0, the conformance of the bitstream cannot be verified without provision of the NAL HRD parameters and all buffering period SEI messages, and, when vcl_hrd_parameters_present_flag is also equal to 0, all picture timing and decoding unit information SEI messages, by some means not specified in this Specification.

The variable NalHrdBpPresentFlag is derived as follows:

– If one or more of the following conditions are true, the value of NalHrdBpPresentFlag is set equal to 1:
  – nal_hrd_parameters_present_flag is present in the bitstream and is equal to 1.
  – The need for the presence of buffering period parameters for NAL HRD operation in the bitstream in buffering period SEI messages is determined by the application, by some means not specified in this Specification.
– Otherwise, the value of NalHrdBpPresentFlag is set equal to 0.

vcl_hrd_parameters_present_flag equal to 1 specifies that VCL HRD parameters (pertaining to the Type I bitstream conformance point) are present in the hrd_parameters( ) syntax structure. vcl_hrd_parameters_present_flag equal to 0 specifies that VCL HRD parameters are not present in the hrd_parameters( ) syntax structure.

NOTE 2 – When vcl_hrd_parameters_present_flag is equal to 0, the conformance of the bitstream cannot be verified without provision of the VCL HRD parameters and all buffering period SEI messages, and, when nal_hrd_parameters_present_flag is also equal to 0, all picture timing and decoding unit information SEI messages, by some means not specified in this Specification.

The variable VclHrdBpPresentFlag is derived as follows:

– If one or more of the following conditions are true, the value of VclHrdBpPresentFlag is set equal to 1:
  – vcl_hrd_parameters_present_flag is present in the bitstream and is equal to 1.
  – The need for the presence of buffering period parameters for VCL HRD operation in the bitstream in buffering period SEI messages is determined by the application, by some means not specified in this Specification.
– Otherwise, the value of VclHrdBpPresentFlag is set equal to 0.

The variable CpbDpbDelaysPresentFlag is derived as follows:

– If one or more of the following conditions are true, the value of CpbDpbDelaysPresentFlag is set equal to 1:
  – nal_hrd_parameters_present_flag is present in the bitstream and is equal to 1.
  – vcl_hrd_parameters_present_flag is present in the bitstream and is equal to 1.
  – The need for presence of CPB and DPB output delays to be present in the bitstream in picture timing SEI messages is determined by the application, by some means not specified in this Specification.
– Otherwise, the value of CpbDpbDelaysPresentFlag is set equal to 0.

When NalHrdBpPresentFlag and VclHrdBpPresentFlag are both equal to 0, the value of CpbDpbDelaysPresentFlag shall be equal to 0.

sub_pic_hrd_params_present_flag equal to 1 specifies that sub-picture level HRD parameters are present and the HRD may operate at access unit level or sub-picture level. sub_pic_hrd_params_present_flag equal to 0 specifies that sub-picture level HRD parameters are not present and the HRD operates at access unit level. When sub_pic_hrd_params_present_flag is not present, its value is inferred to be equal to 0.

tick_divisor_minus2 is used to specify the clock sub-tick. A clock sub-tick is the minimum interval of time that can be represented in the coded data when sub_pic_hrd_params_present_flag is equal to 1.

du_cpb_removal_delay_increment_length_minus1 plus 1 specifies the length, in bits, of the du_cpb_removal_delay_increment_minus1[i] and du_common_cpb_removal_delay_increment_minus1 syntax elements of the picture timing SEI message and the du_spt_cpb_removal_delay_increment syntax element in the decoding unit information SEI message.

sub_pic_cpb_params_in_pic_timing_sei_flag equal to 1 specifies that sub-picture level CPB removal delay parameters are present in picture timing SEI messages and no decoding unit information SEI message is available (in the CVS or provided through external means not specified in this Specification). sub_pic_cpb_params_in_pic_timing_sei_flag equal to 0 specifies that sub-picture level CPB removal delay parameters are present in decoding unit information SEI messages and picture timing SEI messages do not include sub-picture level CPB removal delay parameters. When the sub_pic_cpb_params_in_pic_timing_sei_flag syntax element is not present, it is inferred to be equal to 0.

dpb_output_delay_du_length_minus1 plus 1 specifies the length, in bits, of the pic_dpb_output_du_delay syntax element in the picture timing SEI message and the pic_spt_dpb_output_du_delay syntax element in the decoding unit information SEI message.

bit_rate_scale (together with bit_rate_value_minus1[i]) specifies the maximum input bit rate of the i-th CPB.

cpb_size_scale (together with cpb_size_value_minus1[i]) specifies the CPB size of the i-th CPB when the CPB operates at the access unit level.

cpb_size_du_scale (together with cpb_size_du_value_minus1[i]) specifies the CPB size of the i-th CPB when the CPB operates at sub-picture level.

initial_cpb_removal_delay_length_minus1 plus 1 specifies the length, in bits, of the nal_initial_cpb_removal_delay[i], nal_initial_cpb_removal_offset[i], vcl_initial_cpb_removal_delay[i] and vcl_initial_cpb_removal_offset[i] syntax elements of the buffering period SEI message. When the initial_cpb_removal_delay_length_minus1 syntax element is not present, it is inferred to be equal to 23.

au_cpb_removal_delay_length_minus1 plus 1 specifies the length, in bits, of the cpb_delay_offset syntax element in the buffering period SEI message and the au_cpb_removal_delay_minus1 syntax element in the picture timing SEI message. When the au_cpb_removal_delay_length_minus1 syntax element is not present, it is inferred to be equal to 23.

dpb_output_delay_length_minus1 plus 1 specifies the length, in bits, of the dpb_delay_offset syntax element in the buffering period SEI message and the pic_dpb_output_delay syntax element in the picture timing SEI message. When the dpb_output_delay_length_minus1 syntax element is not present, it is inferred to be equal to 23.

fixed_pic_rate_general_flag[i] equal to 1 indicates that, when HighestTid is equal to i, the temporal distance between the HRD output times of consecutive pictures in output order is constrained as specified below. fixed_pic_rate_general_flag[i] equal to 0 indicates that this constraint may not apply.

When fixed_pic_rate_general_flag[i] is not present, it is inferred to be equal to 0.

fixed_pic_rate_within_cvs_flag[i] equal to 1 indicates that, when HighestTid is equal to i, the temporal distance between the HRD output times of consecutive pictures in output order is constrained as specified below. fixed_pic_rate_within_cvs_flag[i] equal to 0 indicates that this constraint may not apply.

When fixed_pic_rate_general_flag[i] is equal to 1, the value of fixed_pic_rate_within_cvs_flag[i] is inferred to be equal to 1.

elemental_duration_in_tc_minus1[i] plus 1 (when present) specifies, when HighestTid is equal to i, the temporal distance, in clock ticks, between the elemental units that specify the HRD output times of consecutive pictures in output order as specified below. The value of elemental_duration_in_tc_minus1[i] shall be in the range of 0 to 2 047, inclusive.

For each picture n that is output and not the last picture in the bitstream (in output order) that is output, the value of the variable DpbOutputElementalInterval[n] is specified by:

$$\text{DpbOutputElementalInterval[n]} = \text{DpbOutputInterval[n]} \div \text{DeltaToDivisor} \qquad \text{(E-76)}$$

where DpbOutputInterval[n] is specified in Equation C-16 and DeltaToDivisor is specified in Table E.7 based on the value of frame_field_info_present_flag and pic_struct for the CVS containing picture n. Entries marked "-" in Table E.7 indicate a lack of dependence of DeltaToDivisor on the corresponding syntax element.

When HighestTid is equal to i and fixed_pic_rate_general_flag[i] is equal to 1 for a CVS containing picture n, the value computed for DpbOutputElementalInterval[n] shall be equal to ClockTick * (elemental_duration_in_tc_minus1[i] + 1), wherein ClockTick is as specified in Equation C-1 (using the value of ClockTick for the CVS containing picture n) when one of the following conditions is true for the following picture in output order nextPicInOutputOrder that is specified for use in Equation C-16:

– picture nextPicInOutputOrder is in the same CVS as picture n.

– picture nextPicInOutputOrder is in a different CVS and fixed_pic_rate_general_flag[i] is equal to 1 in the CVS containing picture nextPicInOutputOrder, the value of ClockTick is the same for both CVSs and the value of elemental_duration_in_tc_minus1[i] is the same for both CVSs.

When HighestTid is equal to i and fixed_pic_rate_within_cvs_flag[i] is equal to 1 for a CVS containing picture n, the value computed for DpbOutputElementalInterval[n] shall be equal to ClockTick * (elemental_duration_in_tc_minus1[i] + 1), wherein ClockTick is as specified in Equation C-1 (using the value of ClockTick for the CVS containing picture n) when the following picture in output order nextPicInOutputOrder that is specified for use in Equation C-16 is in the same CVS as picture n.

Table E.7 – Divisor for computation of DpbOutputElementalInterval[n]

frame_field_info_present_flag	pic_struct	DeltaToDivisor
0	-	1
1	1	1
1	2	1
1	0	1
1	3	2
1	4	2
1	5	3
1	6	3
1	7	2
1	8	3
1	9	1
1	10	1
1	11	1
1	12	1

low_delay_hrd_flag[i] specifies the HRD operational mode, when HighestTid is equal to i, as specified in Annex C or clause F.13. When not present, the value of low_delay_hrd_flag[i] is inferred to be equal to 0.

NOTE 3 – When low_delay_hrd_flag[i] is equal to 1, "big pictures" that violate the nominal CPB removal times due to the number of bits used by an access unit are permitted. It is expected, but not required, that such "big pictures" occur only occasionally.

cpb_cnt_minus1[i] plus 1 specifies the number of alternative CPB specifications in the bitstream of the CVS when HighestTid is equal to i. The value of cpb_cnt_minus1[i] shall be in the range of 0 to 31, inclusive. When not present, the value of cpb_cnt_minus1[i] is inferred to be equal to 0.

E.3.3 Sub-layer HRD parameters semantics

The variable CpbCnt is set equal to cpb_cnt_minus1[subLayerId] + 1.

bit_rate_value_minus1[i] (together with bit_rate_scale) specifies the maximum input bit rate for the i-th CPB when the CPB operates at the access unit level. bit_rate_value_minus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. For any i > 0, bit_rate_value_minus1[i] shall be greater than bit_rate_value_minus1[i − 1].

When SubPicHrdFlag is equal to 0, the bit rate in bits per second is given by:

$$\text{BitRate[i]} = (\text{ bit\_rate\_value\_minus1[i]} + 1) * 2^{(6 + \text{bit\_rate\_scale})} \tag{E-77}$$

When SubPicHrdFlag is equal to 0 and the bit_rate_value_minus1[i] syntax element is not present, the value of BitRate[i] is inferred to be equal to BrVclFactor * MaxBR for VCL HRD parameters and to be equal to BrNalFactor * MaxBR for NAL HRD parameters, where MaxBR, BrVclFactor and BrNalFactor are specified in clause A.4.

cpb_size_value_minus1[i] is used together with cpb_size_scale to specify the i-th CPB size when the CPB operates at the access unit level. cpb_size_value_minus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. For any i greater than 0, cpb_size_value_minus1[i] shall be less than or equal to cpb_size_value_minus1[i − 1].

When SubPicHrdFlag is equal to 0, the CPB size in bits is given by:

$$\text{CpbSize[i]} = (\text{ cpb\_size\_value\_minus1[i]} + 1) * 2^{(4 + \text{cpb\_size\_scale})} \tag{E-78}$$

When SubPicHrdFlag is equal to 0 and the cpb_size_value_minus1[i] syntax element is not present, the value of CpbSize[i] is inferred to be equal to CpbVclFactor * MaxCPB for VCL HRD parameters and to be equal to CpbNalFactor * MaxCPB for NAL HRD parameters, where MaxCPB, CpbVclFactor and CpbNalFactor are specified in clause A.4.

cpb_size_du_value_minus1[i] is used together with cpb_size_du_scale to specify the i-th CPB size when the CPB operates at sub-picture level. cpb_size_du_value_minus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. For any i greater than 0, cpb_size_du_value_minus1[i] shall be less than or equal to cpb_size_du_value_minus1[i − 1].

When SubPicHrdFlag is equal to 1, the CPB size in bits is given by:

$$\text{CpbSize[i]} = (\text{ cpb\_size\_du\_value\_minus1[i]} + 1) * 2^{(4 + \text{cpb\_size\_du\_scale})} \tag{E-79}$$

When SubPicHrdFlag is equal to 1 and the cpb_size_du_value_minus1[i] syntax element is not present, the value of CpbSize[i] is inferred to be equal to CpbVclFactor * MaxCPB for VCL HRD parameters and to be equal to CpbNalFactor * MaxCPB for NAL HRD parameters, where MaxCPB, CpbVclFactor and CpbNalFactor are specified in clause A.4.

bit_rate_du_value_minus1[i] (together with bit_rate_scale) specifies the maximum input bit rate for the i-th CPB when the CPB operates at the sub-picture level. bit_rate_du_value_minus1[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. For any i > 0, bit_rate_du_value_minus1[i] shall be greater than bit_rate_du_value_minus1[i − 1].

When SubPicHrdFlag is equal to 1, the bit rate in bits per second is given by:

$$\text{BitRate[i]} = (\text{ bit\_rate\_du\_value\_minus1[i]} + 1) * 2^{(6 + \text{bit\_rate\_scale})} \tag{E-80}$$

When SubPicHrdFlag is equal to 1 and the bit_rate_du_value_minus1[i] syntax element is not present, the value of BitRate[i] is inferred to be equal to BrVclFactor * MaxBR for VCL HRD parameters and to be equal to BrNalFactor * MaxBR for NAL HRD parameters, where MaxBR, BrVclFactor and BrNalFactor are specified in clause A.4.

cbr_flag[i] equal to 0 specifies that to decode this CVS by the HRD using the i-th CPB specification, the hypothetical stream scheduler (HSS) operates in an intermittent bit rate mode. cbr_flag[i] equal to 1 specifies that the HSS operates in a constant bit rate (CBR) mode. When not present, the value of cbr_flag[i] is inferred to be equal to 0.

Annex F

Common specifications for multi-layer extensions

(This annex forms an integral part of this Recommendation | International Standard.)

F.1 Scope

This annex specifies the common syntax, semantics and decoding processes for multi-layer video coding extensions, with references made to clauses 2-9 and Annexes A-E, G and H.

F.2 Normative references

The list of normative references in clause 2 applies.

F.3 Definitions

For the purpose of this annex, the following definitions apply in addition to the definitions in clause 3. These definitions are either not present in clause 3 or replace definitions in clause 3.

F.3.1 **access unit**: A set of *NAL units* that are associated with each other according to a specified classification rule, are consecutive in *decoding order* and contain at most one *coded picture* with any specific value of nuh_layer_id.

F.3.2 **additional layer set**: A set of *layers* of a *bitstream* with a set of *layers* of one or more *non-base layer subtrees*.

F.3.3 **alternative output layer**: A *layer* that is a *reference layer* of an *output layer* and which may include a *picture* that may be output when no picture of the *output layer* is present in the *access unit* containing the *picture*.

F.3.4 **associated IRAP picture**: The previous *IRAP picture* in *decoding order* within the same *layer* (if present).

F.3.5 **auxiliary picture**: A *picture* that has no normative effect on the *decoding process* of *primary pictures* and with a nuh_layer_id value such that AuxId[nuh_layer_id] is greater than 0.

F.3.6 **auxiliary picture layer**: A *layer* with a nuh_layer_id value such that AuxId[nuh_layer_id] is greater than 0.

F.3.7 **base bitstream partition**: A *bitstream partition* that contains the layer with nuh_layer_id equal to SmallestLayerId.

F.3.8 **bitstream partition**: A sequence of bits, in the form of a *NAL unit stream* or a *byte stream*, that is a subset of a *bitstream* according to a *partitioning scheme*.

F.3.9 **broken link access (BLA) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to SmallestLayerId is a *BLA picture*.

F.3.10 **clean random access (CRA) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to SmallestLayerId is a *CRA picture*.

F.3.11 **coded layer-wise video sequence (CLVS)**: A sequence of *coded pictures*, with the same nuh_layer_id value nuhLayerId, that consists, in decoding order, of an *IRAP picture* with NoRaslOutputFlag equal to 1 or a *picture* with FirstPicInLayerDecodedFlag[nuhLayerId] equal to 0, followed by all *coded pictures*, if any, up to but excluding the next *IRAP picture* with NoRaslOutputFlag equal to 1 or the next *picture* with FirstPicInLayerDecodedFlag[nuhLayerId] equal to 0.

F.3.12 **coded picture**: A *coded representation* of a *picture* comprising *VCL NAL units* with a particular value of nuh_layer_id within an *access unit* and containing all *CTUs* of the *picture*.

F.3.13 **coded picture buffer (CPB)**: A first-in first-out buffer containing *decoding units* in *decoding order* specified in the *hypothetical reference decoder* in Annex C or in clause F.13.

F.3.14 **coded video sequence (CVS)**: A sequence of *access units* that consists, in decoding order, of an *initial IRAP access unit*, followed by zero or more *access units* that are not *initial IRAP access units*, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *initial IRAP access unit*.

F.3.15 **collocated sample**: A reference picture sample located at a corresponding position to the current sample as determined through the resampling process, for use in *inter-layer prediction*.

F.3.16 **cross-layer random access skip (CL-RAS) picture**: A *picture* with nuh_layer_id equal to layerId such that LayerInitializedFlag[layerId] is equal to 0 when the decoding process for starting the decoding of a coded picture with nuh_layer_id greater than 0 is invoked.

F.3.17 **decoded picture buffer (DPB)**: A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C or in clause F.13.

F.3.18 **decoding unit**: A *partition unit* if SubPicHrdFlag is equal to 0 or a subset of a *partition unit* otherwise, consisting of one or more *VCL NAL units* in a *partition unit* and the *associated non-VCL NAL units*.

F.3.19 **direct predicted layer**: A *layer* for which another *layer* is a *direct reference layer*.

F.3.20 **direct reference layer**: A *layer* that may be used for *inter-layer prediction* of another *layer*.

F.3.21 **direct reference layer picture**: A *picture* in a *direct reference layer* that is used as input to derive an *inter-layer reference picture* for *inter-layer prediction* of the current *picture* and is in the same *access unit* as the current *picture*.

F.3.22 **independent layer**: A *layer* that does not have *direct reference layers*.

F.3.23 **indirect predicted layer**: A *layer* for which another *layer* is an *indirect reference layer*.

F.3.24 **indirect reference layer**: A *layer* that is not a *direct reference layer* of a second *layer* but is a *direct reference layer* of a third *layer* that is a *reference layer* of the second *layer*.

F.3.25 **initial intra random access point (IRAP) access unit**: An *IRAP access unit* in which the *coded picture* with nuh_layer_id equal to SmallestLayerId has NoRaslOutputFlag equal to 1.

F.3.26 **inter-layer prediction**: A *prediction* in a manner that is dependent on data elements (e.g., sample values or motion vectors) of *reference pictures* with a different value of nuh_layer_id than that of the current *picture*.

F.3.27 **inter-layer reference picture**: A *reference picture* that may be used for *inter-layer prediction* of the current *picture* and is in the same *access unit* as the current *picture*.

F.3.28 **intra random access point (IRAP) access unit**: An *access unit* in which the *coded picture* with nuh_layer_id equal to SmallestLayerId is an *IRAP picture*.

F.3.29 **intra random access point (IRAP) picture**: A coded *picture* for which each *VCL NAL unit* has nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.

NOTE – An IRAP picture belonging to an independent layer contains only I slices. An IRAP picture belonging to a predicted layer with nuh_layer_id value currLayerId may contain P, B and I slices, cannot use inter prediction from other pictures with nuh_layer_id equal to currLayerId and may use inter-layer prediction from its direct reference layers. The IRAP picture belonging to a predicted layer with nuh_layer_id value currLayerId and all subsequent non-RASL pictures with nuh_layer_id equal to currLayerId in decoding order can be correctly decoded without performing the decoding process of any pictures with nuh_layer_id equal to currLayerId that precede the IRAP picture in decoding order, when the necessary parameter sets are available when they need to be activated and LayerInitializedFlag[refLayerId] is equal to 1 for refLayerId equal to all nuh_layer_id values of the direct reference layers of the layer with nuh_layer_id equal to currLayerId.

F.3.30 **layer subtree**: A subset of the *layers* of a *layer tree* including all the *reference layers* of the *layers* within the subset.

F.3.31 **layer tree**: A set of *layers* such that each *layer* in the set of *layers* is a *predicted layer* or a *reference layer* of at least one other *layer* in the set of *layers* and no *layer* outside the set of *layers* is a *predicted layer* or a *reference layer* of any *layer* in the set of *layers*.

F.3.32 **leading picture**: A *picture* that is in the same *layer* as the *associated IRAP picture* and precedes the *associated IRAP picture* in *output order*.

F.3.33 **necessary layer**: A *layer* in an *output operation point* associated with an *OLS*, the layer being an *output layer* of the *OLS*, or a *reference layer* of an *output layer* of the *OLS*.

F.3.34 **non-base layer**: A *layer* in which all *VCL NAL units* have the same nuh_layer_id value greater than 0.

F.3.35 **non-base layer subtree**: A *layer subtree* that does not include the *base layer*.

F.3.36 **output layer**: A *layer* of an *output layer set* that is output when TargetOlsIdx is equal to the index of the *output layer set*.

F.3.37 **output layer set (OLS)**: A set of *layers* consisting of the *layers* of one of the specified *layer sets*, where one or more *layers* in the set of *layers* are indicated to be *output layers*.

F.3.38 **output operation point**: A *bitstream* that is created from an input *bitstream* by operation of the *sub-bitstream extraction process* with the input *bitstream*, a target highest TemporalId and a target layer identifier list as inputs, and that is associated with a set of *output layers*.

F.3.39 **output order**: The order in which the *decoded pictures* are output from the *decoded picture buffer* (for the *decoded pictures* that are to be output from the *decoded picture buffer*), or the order in which the *access units* are considered to be output from the *decoded picture buffer* (for the *access units* the decoding of which results into *decoded pictures* that are to be output from the *decoded picture buffer*).

F.3.40 **partitioning scheme**: A *partitioning* of *layers* in an *OLS* into one or more *bitstream partitions* such that each *layer* in the *OLS* is included in exactly one *bitstream partition* of the partitioning scheme and each *bitstream partition* of the partitioning scheme contains one or more *layers*.

F.3.41 **partition unit**: A subset of *access unit* consisting of the *picture units* of the *layers* in a *bitstream partition*.

F.3.42 **picture order count (POC)**: A variable that is associated with each *picture* and that, at any moment, uniquely identifies the associated *picture* among all *pictures* with the same value of nuh_layer_id in the CVS, and, when the associated *picture* is to be output from the *decoded picture buffer*, indicates the position of the associated *picture* in *output order* relative to the *output order* positions of the other *pictures* with the same value of nuh_layer_id in the same *CVS* that are to be output from the *decoded picture buffer*.

F.3.43 **picture order count (POC) resetting period**: A sequence of *access units* in *decoding order*, starting with an *access unit* with poc_reset_idc equal to 1 or 2 and a particular value of poc_reset_period_id and including all *access units* that either have the same value of poc_reset_period_id or have poc_reset_idc equal to 0.

F.3.44 **picture order count (POC) resetting picture**: A *picture* that is the first *picture*, in *decoding order*, of a *layer* of a *POC resetting period*.

F.3.45 **predicted layer**: A *layer* that is a *direct predicted layer* or an *indirect predicted layer* of another *layer*.

F.3.46 **primary picture**: A *picture* with nuh_layer_id value such that AuxId[nuh_layer_id] is equal to 0.

F.3.47 **primary picture layer**: A *layer* with a nuh_layer_id value such that AuxId[nuh_layer_id] is equal to 0.

F.3.48 **quality scalability**: A type of scalability, represented by a scalability dimension, that indicates that the *coded pictures* in all *layers* of a bitstream have the same values of *luma* width, *luma* height, *chroma* width and *chroma* height, but may have different fidelity level or different values of *luma* or *chroma* bit depths.

F.3.49 **reference layer**: A *layer* that is a *direct reference layer* or an *indirect reference layer* of another *layer*.

F.3.50 **reference picture list**: A list of *reference pictures* that is used for *inter prediction* or *inter-layer prediction* of a *P* or *B slice*.

F.3.51 **scalability dimension**: An indication of the type of *non-base layers* which may be present in a *CVS*.

F.3.52 **source picture for inter-layer prediction**: A *decoded picture* that either is, or is used in deriving, an *inter-layer reference picture* that is included in an inter-layer reference picture set of the current *picture*.

F.3.53 **spatial scalability**: A type of scalability, represented by a scalability dimension, that indicates that the *coded pictures* in different *layers* of a *bitstream* have different values of *luma* or *chroma* width or height.

F.3.54 **sub-bitstream extraction process**: A specified process by which *NAL units* in a *bitstream* that do not belong to a target set, determined by a target highest TemporalId and a target *layer identifier list*, are removed from the *bitstream*, with the output sub-bitstream consisting of the NAL units in the *bitstream* that belong to the target set.

NOTE – Depending on whether the target layer identifier list includes nuh_layer_id equal to 0 and on the value of vps_base_layer_internal_flag, the sub-bitstream extraction process may refer to the process specified in clause 10, clause F.10.1 or clause F.10.3.

F.3.55 **target output layer set**: The *output layer set* for which the index is equal to TargetOlsIdx.

F.3.56 **trailing picture**: A *picture* that is in the same *layer* as the *associated IRAP picture* and follows the *associated IRAP picture* in *output order*.

F.3.57 **view**: A sequence of *pictures* associated with the same value of ViewOrderIdx.

NOTE – A view typically represents a sequence of pictures captured by one camera.

F.4 Abbreviations

For the purpose of this annex, the following abbreviations apply in addition to the abbreviations in clause 4:

BPB Bitstream Partition Buffer

HBPS Hypothetical Bitstream Partition Scheduler

OLS Output Layer Set

F.5 Conventions

The specifications in clause 5 and its subclauses apply.

F.6 Bitstream and picture formats, partitionings, scanning processes and neighbouring relationships

The specifications in clause 6 and its subclauses apply.

F.7 Syntax and semantics

F.7.1 Method of specifying syntax in tabular form

The specifications in clause 7.1 apply.

F.7.2 Specification of syntax functions, categories and descriptors

The specifications in clause 7.2 apply, with the following additions:

more_data_in_slice_segment_header_extension( ) is specified as follows:

– If (the current position in the slice_segment_header( ) syntax structure) − (the position immediately following slice_segment_header_extension_length) is less than (slice_segment_header_extension_length * 8), the return value of more_data_in_slice_segment_header_extension( ) is equal to TRUE.

– Otherwise, the return value of more_data_in_slice_segment_header_extension( ) is equal to FALSE.

F.7.3 Syntax in tabular form

F.7.3.1 NAL unit syntax

F.7.3.1.1 General NAL unit syntax

The specifications in clause 7.3.1.1 apply.

F.7.3.1.2 NAL unit header syntax

The specifications in clause 7.3.1.2 apply.

F.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

F.7.3.2.1 Video parameter set RBSP

video_parameter_set_rbsp( ) {	**Descriptor**
vps_video_parameter_set_id	u(4)
vps_base_layer_internal_flag	u(1)
vps_base_layer_available_flag	u(1)
vps_max_layers_minus1	u(6)
vps_max_sub_layers_minus1	u(3)
vps_temporal_id_nesting_flag	u(1)
vps_reserved_0xffff_16bits	u(16)
profile_tier_level(1, vps_max_sub_layers_minus1)	
vps_sub_layer_ordering_info_present_flag	u(1)
for(i = (vps_sub_layer_ordering_info_present_flag ? 0 : vps_max_sub_layers_minus1); i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering_minus1[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase_plus1[i]	ue(v)
}	
vps_max_layer_id	u(6)
vps_num_layer_sets_minus1	ue(v)
for(i = 1; i <= vps_num_layer_sets_minus1; i++)	

for(j = 0; j <= vps_max_layer_id; j++)	
layer_id_included_flag[i][j]	u(1)
vps_timing_info_present_flag	u(1)
if(vps_timing_info_present_flag) {	
vps_num_units_in_tick	u(32)
vps_time_scale	u(32)
vps_poc_proportional_to_timing_flag	u(1)
if(vps_poc_proportional_to_timing_flag)	
vps_num_ticks_poc_diff_one_minus1	ue(v)
vps_num_hrd_parameters	ue(v)
for(i = 0; i < vps_num_hrd_parameters; i++) {	
hrd_layer_set_idx[i]	ue(v)
if(i > 0)	
cprms_present_flag[i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
}	
vps_extension_flag	u(1)
if(vps_extension_flag) {	
while(!byte_aligned( ))	
vps_extension_alignment_bit_equal_to_one	u(1)
vps_extension( )	
vps_extension2_flag	u(1)
if(vps_extension2_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

F.7.3.2.1.1 Video parameter set extension syntax

vps_extension( ) {	**Descriptor**
if(vps_max_layers_minus1 > 0 && vps_base_layer_internal_flag)	
profile_tier_level(0, vps_max_sub_layers_minus1)	
splitting_flag	u(1)
for(i = 0, NumScalabilityTypes = 0; i < 16; i++) {	
scalability_mask_flag[i]	u(1)
NumScalabilityTypes += scalability_mask_flag[i]	
}	
for(j = 0; j < (NumScalabilityTypes − splitting_flag); j++)	
dimension_id_len_minus1[j]	u(3)
vps_nuh_layer_id_present_flag	u(1)
for(i = 1; i <= MaxLayersMinus1; i++) {	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh[i]	u(6)
if(!splitting_flag)	

for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v)
}	
view_id_len	u(4)
if(view_id_len > 0)	
for(i = 0; i < NumViews; i++)	
view_id_val[i]	u(v)
for(i = 1; i <= MaxLayersMinus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
if(NumIndependentLayers > 1)	
num_add_layer_sets	ue(v)
for(i = 0; i < num_add_layer_sets; i++)	
for(j = 1; j < NumIndependentLayers; j++)	
highest_layer_idx_plus1[i][j]	u(v)
vps_sub_layers_max_minus1_present_flag	u(1)
if(vps_sub_layers_max_minus1_present_flag)	
for(i = 0; i <= MaxLayersMinus1; i++)	
sub_layers_vps_max_minus1[i]	u(3)
max_tid_ref_present_flag	u(1)
if(max_tid_ref_present_flag)	
for(i = 0; i < MaxLayersMinus1; i++)	
for(j = i + 1; j <= MaxLayersMinus1; j++)	
if(direct_dependency_flag[j][i])	
max_tid_il_ref_pics_plus1[i][j]	u(3)
default_ref_layers_active_flag	u(1)
vps_num_profile_tier_level_minus1	ue(v)
for(i = vps_base_layer_internal_flag ? 2 : 1; i <= vps_num_profile_tier_level_minus1; i++) {	
vps_profile_present_flag[i]	u(1)
profile_tier_level(vps_profile_present_flag[i], vps_max_sub_layers_minus1)	
}	
if(NumLayerSets > 1) {	
num_add_olss	ue(v)
default_output_layer_idc	u(2)
}	
NumOutputLayerSets = num_add_olss + NumLayerSets	
for(i = 1; i < NumOutputLayerSets; i++) {	
if(NumLayerSets > 2 && i >= NumLayerSets)	
layer_set_idx_for_ols_minus1[i]	u(v)
if(i > vps_num_layer_sets_minus1 || defaultOutputLayerIdc = = 2)	
for(j = 0; j < NumLayersInIdList[OlsIdxToLsIdx[i]]; j++)	
output_layer_flag[i][j]	u(1)
for(j = 0; j < NumLayersInIdList[OlsIdxToLsIdx[i]]; j++)	
if(NecessaryLayerFlag[i][j] && vps_num_profile_tier_level_minus1 > 0)	
profile_tier_level_idx[i][j]	u(v)
if(NumOutputLayersInOutputLayerSet[i] = = 1 && NumDirectRefLayers[OlsHighestOutputLayerId[i]] > 0)	
alt_output_layer_flag[i]	u(1)

}	
vps_num_rep_formats_minus1	ue(v)
for(i = 0; i <= vps_num_rep_formats_minus1; i++)	
rep_format( )	
if(vps_num_rep_formats_minus1 > 0)	
rep_format_idx_present_flag	u(1)
if(rep_format_idx_present_flag)	
for(i = vps_base_layer_internal_flag ? 1 : 0; i <= MaxLayersMinus1; i++)	
vps_rep_format_idx[i]	u(v)
max_one_active_ref_layer_flag	u(1)
vps_poc_lsb_aligned_flag	u(1)
for(i = 1; i <= MaxLayersMinus1; i++)	
if(NumDirectRefLayers[layer_id_in_nuh[i]] = = 0)	
poc_lsb_not_present_flag[i]	u(1)
dpb_size( )	
direct_dep_type_len_minus2	ue(v)
direct_dependency_all_layers_flag	u(1)
if(direct_dependency_all_layers_flag)	
direct_dependency_all_layers_type	u(v)
else {	
for(i = vps_base_layer_internal_flag ? 1 : 2; i <= MaxLayersMinus1; i++)	
for(j = vps_base_layer_internal_flag ? 0 : 1; j < i; j++)	
if(direct_dependency_flag[i][j])	
direct_dependency_type[i][j]	u(v)
}	
vps_non_vui_extension_length	ue(v)
for(i = 1; i <= vps_non_vui_extension_length; i++)	
vps_non_vui_extension_data_byte	u(8)
vps_vui_present_flag	u(1)
if(vps_vui_present_flag) {	
while(!byte_aligned( ))	
vps_vui_alignment_bit_equal_to_one	u(1)
vps_vui( )	
}	
}	

F.7.3.2.1.2 Representation format syntax

rep_format( ) {	**Descriptor**
pic_width_vps_in_luma_samples	u(16)
pic_height_vps_in_luma_samples	u(16)
chroma_and_bit_depth_vps_present_flag	u(1)
if(chroma_and_bit_depth_vps_present_flag) {	
chroma_format_vps_idc	u(2)
if(chroma_format_vps_idc = = 3)	
separate_colour_plane_vps_flag	u(1)
bit_depth_vps_luma_minus8	u(4)
bit_depth_vps_chroma_minus8	u(4)
}	
conformance_window_vps_flag	u(1)
if(conformance_window_vps_flag) {	
conf_win_vps_left_offset	ue(v)
conf_win_vps_right_offset	ue(v)
conf_win_vps_top_offset	ue(v)
conf_win_vps_bottom_offset	ue(v)
}	
}	

F.7.3.2.1.3 DPB size syntax

dpb_size( ) {	**Descriptor**
for(i = 1; i < NumOutputLayerSets; i++) {	
currLsIdx = OlsIdxToLsIdx[i]	
sub_layer_flag_info_present_flag[i]	u(1)
for(j = 0; j <= MaxSubLayersInLayerSetMinus1[currLsIdx]; j++) {	
if(j > 0 && sub_layer_flag_info_present_flag[i])	
sub_layer_dpb_info_present_flag[i][j]	u(1)
if(sub_layer_dpb_info_present_flag[i][j]) {	
for(k = 0; k < NumLayersInIdList[currLsIdx]; k++)	
if(NecessaryLayerFlag[i][k] && (vps_base_layer_internal_flag || (LayerSetLayerIdList[currLsIdx][k] != 0)))	
max_vps_dec_pic_buffering_minus1[i][k][j]	ue(v)
max_vps_num_reorder_pics[i][j]	ue(v)
max_vps_latency_increase_plus1[i][j]	ue(v)
}	
}	
}	
}	

F.7.3.2.1.4 VPS VUI syntax

vps_vui( ) {	**Descriptor**
cross_layer_pic_type_aligned_flag	u(1)
if(!cross_layer_pic_type_aligned_flag)	
cross_layer_irap_aligned_flag	u(1)
if(cross_layer_irap_aligned_flag)	
all_layers_idr_aligned_flag	u(1)
bit_rate_present_vps_flag	u(1)
pic_rate_present_vps_flag	u(1)
if(bit_rate_present_vps_flag || pic_rate_present_vps_flag)	
for(i = vps_base_layer_internal_flag ? 0 : 1; i < NumLayerSets; i++)	
for(j = 0; j <= MaxSubLayersInLayerSetMinus1[i]; j++) {	
if(bit_rate_present_vps_flag)	
bit_rate_present_flag[i][j]	u(1)
if(pic_rate_present_vps_flag)	
pic_rate_present_flag[i][j]	u(1)
if(bit_rate_present_flag[i][j]) {	
avg_bit_rate[i][j]	u(16)
max_bit_rate[i][j]	u(16)
}	
if(pic_rate_present_flag[i][j]) {	
constant_pic_rate_idc[i][j]	u(2)
avg_pic_rate[i][j]	u(16)
}	
}	
video_signal_info_idx_present_flag	u(1)
if(video_signal_info_idx_present_flag)	
vps_num_video_signal_info_minus1	u(4)
for(i = 0; i <= vps_num_video_signal_info_minus1; i++)	
video_signal_info( )	
if(video_signal_info_idx_present_flag && vps_num_video_signal_info_minus1 > 0)	
for(i = vps_base_layer_internal_flag ? 0 : 1; i <= MaxLayersMinus1; i++)	
vps_video_signal_info_idx[i]	u(4)
tiles_not_in_use_flag	u(1)
if(!tiles_not_in_use_flag) {	
for(i = vps_base_layer_internal_flag ? 0 : 1; i <= MaxLayersMinus1; i++) {	
tiles_in_use_flag[i]	u(1)
if(tiles_in_use_flag[i])	
loop_filter_not_across_tiles_flag[i]	u(1)
}	
for(i = vps_base_layer_internal_flag ? 1 : 2; i <= MaxLayersMinus1; i++)	
for(j = 0; j < NumDirectRefLayers[layer_id_in_nuh[i]]; j++) {	
layerIdx = LayerIdxInVps[IdDirectRefLayer[layer_id_in_nuh[i]][j]]	
if(tiles_in_use_flag[i] && tiles_in_use_flag[layerIdx])	
tile_boundaries_aligned_flag[i][j]	u(1)
}	
}	
wpp_not_in_use_flag	u(1)

if(!wpp_not_in_use_flag)	
for(i = vps_base_layer_internal_flag ? 0 : 1; i <= MaxLayersMinus1; i++)	
wpp_in_use_flag[i]	u(1)
single_layer_for_non_irap_flag	u(1)
higher_layer_irap_skip_flag	u(1)
ilp_restricted_ref_layers_flag	u(1)
if(ilp_restricted_ref_layers_flag)	
for(i = 1; i <= MaxLayersMinus1; i++)	
for(j = 0; j < NumDirectRefLayers[layer_id_in_nuh[i]]; j++)	
if(vps_base_layer_internal_flag || IdDirectRefLayer[layer_id_in_nuh[i]][j] > 0) {	
min_spatial_segment_offset_plus1[i][j]	ue(v)
if(min_spatial_segment_offset_plus1[i][j] > 0) {	
ctu_based_offset_enabled_flag[i][j]	u(1)
if(ctu_based_offset_enabled_flag[i][j])	
min_horizontal_ctu_offset_plus1[i][j]	ue(v)
}	
}	
vps_vui_bsp_hrd_present_flag	u(1)
if(vps_vui_bsp_hrd_present_flag)	
vps_vui_bsp_hrd_params( )	
for(i = 1; i <= MaxLayersMinus1; i++)	
if(NumDirectRefLayers[layer_id_in_nuh[i]] = = 0)	
base_layer_parameter_set_compatibility_flag[i]	u(1)
}	

F.7.3.2.1.5 Video signal info syntax

video_signal_info( ) {	**Descriptor**
video_vps_format	u(3)
video_full_range_vps_flag	u(1)
colour_primaries_vps	u(8)
transfer_characteristics_vps	u(8)
matrix_coeffs_vps	u(8)
}	

F.7.3.2.1.6 VPS VUI bitstream partition HRD parameters syntax

vps_vui_bsp_hrd_params( ) {	**Descriptor**
vps_num_add_hrd_params	ue(v)
for(i = vps_num_hrd_parameters; i < vps_num_hrd_parameters + vps_num_add_hrd_params; i++) {	
if(i > 0)	
cprms_add_present_flag[i]	u(1)
num_sub_layer_hrd_minus1[i]	ue(v)
hrd_parameters(cprms_add_present_flag[i], num_sub_layer_hrd_minus1[i])	
}	
if(vps_num_hrd_parameters + vps_num_add_hrd_params > 0)	
for(h = 1; h < NumOutputLayerSets; h++) {	
num_signalled_partitioning_schemes[h]	ue(v)
for(j = 1; j < num_signalled_partitioning_schemes[h] + 1; j++) {	
num_partitions_in_scheme_minus1[h][j]	ue(v)
for(k = 0; k <= num_partitions_in_scheme_minus1[h][j]; k++)	
for(r = 0; r < NumLayersInIdList[OlsIdxToLsIdx[h]]; r++)	
layer_included_in_partition_flag[h][j][k][r]	u(1)
}	
for(i = 0; i < num_signalled_partitioning_schemes[h] + 1; i++)	
for(t = 0; t <= MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[h]]; t++) {	
num_bsp_schedules_minus1[h][i][t]	ue(v)
for(j = 0; j <= num_bsp_schedules_minus1[h][i][t]; j++)	
for(k = 0; k <= num_partitions_in_scheme_minus1[h][i]; k++) {	
if(vps_num_hrd_parameters + vps_num_add_hrd_params > 1)	
bsp_hrd_idx[h][i][t][j][k]	u(v)
bsp_sched_idx[h][i][t][j][k]	ue(v)
}	
}	
}	
}	

F.7.3.2.2 Sequence parameter set RBSP syntax

F.7.3.2.2.1 General sequence parameter set RBSP syntax

seq_parameter_set_rbsp( ) {	**Descriptor**
sps_video_parameter_set_id	u(4)
if(nuh_layer_id = = 0)	
sps_max_sub_layers_minus1	u(3)
else	
sps_ext_or_max_sub_layers_minus1	u(3)
MultiLayerExtSpsFlag = (nuh_layer_id != 0 && sps_ext_or_max_sub_layers_minus1 = = 7)	
if(!MultiLayerExtSpsFlag) {	
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
}	

sps_seq_parameter_set_id	ue(v)
if(MultiLayerExtSpsFlag) {	
update_rep_format_flag	u(1)
if(update_rep_format_flag)	
sps_rep_format_idx	u(8)
} else {	
chroma_format_idc	ue(v)
if(chroma_format_idc = = 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)
if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
}	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
if(!MultiLayerExtSpsFlag) {	
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
}	
log2_min_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_luma_coding_block_size	ue(v)
log2_min_luma_transform_block_size_minus2	ue(v)
log2_diff_max_min_luma_transform_block_size	ue(v)
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enabled_flag	u(1)
if(scaling_list_enabled_flag) {	
if(MultiLayerExtSpsFlag)	
sps_infer_scaling_list_flag	u(1)
if(sps_infer_scaling_list_flag)	
sps_scaling_list_ref_layer_id	u(6)
else {	
sps_scaling_list_data_present_flag	u(1)
if(sps_scaling_list_data_present_flag)	
scaling_list_data( )	
}	
}	

amp_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_sample_bit_depth_luma_minus1	u(4)
pcm_sample_bit_depth_chroma_minus1	u(4)
log2_min_pcm_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_luma_coding_block_size	ue(v)
pcm_loop_filter_disabled_flag	u(1)
}	
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
st_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
if(long_term_ref_pics_present_flag) {	
num_long_term_ref_pics_sps	ue(v)
for(i = 0; i < num_long_term_ref_pics_sps; i++) {	
lt_ref_pic_poc_lsb_sps[i]	u(v)
used_by_curr_pic_lt_sps_flag[i]	u(1)
}	
}	
sps_temporal_mvp_enabled_flag	u(1)
strong_intra_smoothing_enabled_flag	u(1)
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
sps_extension_present_flag	u(1)
if(sps_extension_present_flag) {	
sps_range_extension_flag	u(1)
sps_multilayer_extension_flag	u(1)
sps_3d_extension_flag	u(1)
sps_scc_extension_flag	u(1)
sps_extension_4bits	u(5)
}	
if(sps_range_extension_flag)	
sps_range_extension( )	
if(sps_multilayer_extension_flag)	
sps_multilayer_extension( )	
if(sps_3d_extension_flag)	
sps_3d_extension( ) /* specified in Annex I */	
if(sps_scc_extension_flag)	
sps_scc_extension( )	
if(sps_extension_4bits)	
while(more_rbsp_data( ))	
sps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

F.7.3.2.2.2 Sequence parameter set range extension syntax

The specifications in clause 7.3.2.2.2 apply.

F.7.3.2.2.3 Sequence parameter set screen content coding extension syntax

The specifications in clause 7.3.2.2.3 apply.

F.7.3.2.2.4 Sequence parameter set multilayer extension syntax

sps_multilayer_extension( ) {	**Descriptor**
inter_view_mv_vert_constraint_flag	u(1)
}	

F.7.3.2.3 Picture parameter set RBSP syntax

F.7.3.2.3.1 General picture parameter set RBSP syntax

The specifications in clause 7.3.2.3.1 apply.

F.7.3.2.3.2 Picture parameter set range extension syntax

The specifications in clause 7.3.2.3.2 apply.

F.7.3.2.3.3 Picture parameter set screen content coding extension syntax

The specifications in clause 7.3.2.3.3 apply.

F.7.3.2.3.4 Picture parameter set multilayer extension syntax

pps_multilayer_extension( ) {	**Descriptor**
poc_reset_info_present_flag	u(1)
pps_infer_scaling_list_flag	u(1)
if(pps_infer_scaling_list_flag)	
pps_scaling_list_ref_layer_id	u(6)
num_ref_loc_offsets	ue(v)
for(i = 0; i < num_ref_loc_offsets; i++) {	
ref_loc_offset_layer_id[i]	u(6)
scaled_ref_layer_offset_present_flag[i]	u(1)
if(scaled_ref_layer_offset_present_flag[i]) {	
scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]]	se(v)
scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]]	se(v)
scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]]	se(v)
scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]]	se(v)
}	
ref_region_offset_present_flag[i]	u(1)
if(ref_region_offset_present_flag[i]) {	
ref_region_left_offset[ref_loc_offset_layer_id[i]]	se(v)
ref_region_top_offset[ref_loc_offset_layer_id[i]]	se(v)
ref_region_right_offset[ref_loc_offset_layer_id[i]]	se(v)
ref_region_bottom_offset[ref_loc_offset_layer_id[i]]	se(v)
}	
resample_phase_set_present_flag[i]	u(1)
if(resample_phase_set_present_flag[i]) {	
phase_hor_luma[ref_loc_offset_layer_id[i]]	ue(v)
phase_ver_luma[ref_loc_offset_layer_id[i]]	ue(v)
phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]]	ue(v)
phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]]	ue(v)
}	
}	
colour_mapping_enabled_flag	u(1)
if(colour_mapping_enabled_flag)	
colour_mapping_table( )	
}	

F.7.3.2.3.5 General colour mapping table syntax

colour_mapping_table( ) {	**Descriptor**
num_cm_ref_layers_minus1	ue(v)
for(i = 0; i <= num_cm_ref_layers_minus1; i++)	
cm_ref_layer_id[i]	u(6)
cm_octant_depth	u(2)
cm_y_part_num_log2	u(2)
luma_bit_depth_cm_input_minus8	ue(v)
chroma_bit_depth_cm_input_minus8	ue(v)
luma_bit_depth_cm_output_minus8	ue(v)
chroma_bit_depth_cm_output_minus8	ue(v)
cm_res_quant_bits	u(2)
cm_delta_flc_bits_minus1	u(2)
if(cm_octant_depth = = 1) {	
cm_adapt_threshold_u_delta	se(v)
cm_adapt_threshold_v_delta	se(v)
}	
colour_mapping_octants(0, 0, 0, 0, 1 << cm_octant_depth)	
}	

F.7.3.2.3.6 Colour mapping octants syntax

colour_mapping_octants(inpDepth, idxY, idxCb, idxCr, inpLength) {	**Descriptor**
if(inpDepth < cm_octant_depth)	
split_octant_flag	u(1)
if(split_octant_flag)	
for(k = 0; k < 2; k++)	
for(m = 0; m < 2; m++)	
for(n = 0; n < 2; n++)	
colour_mapping_octants(inpDepth + 1, idxY + PartNumY * k * inpLength / 2, idxCb + m * inpLength / 2, idxCr + n * inpLength / 2, inpLength / 2)	
else	
for(i = 0; i < PartNumY; i++) {	
idxShiftY = idxY + (i << (cm_octant_depth − inpDepth))	
for(j = 0; j < 4; j++) {	
coded_res_flag[idxShiftY][idxCb][idxCr][j]	u(1)
if(coded_res_flag[idxShiftY][idxCb][idxCr][j])	
for(c = 0; c < 3; c++) {	
res_coeff_q[idxShiftY][idxCb][idxCr][j][c]	ue(v)
res_coeff_r[idxShiftY][idxCb][idxCr][j][c]	u(v)
if(res_coeff_q[idxShiftY][idxCb][idxCr][j][c] || res_coeff_r[idxShiftY][idxCb][idxCr][j][c])	
res_coeff_s[idxShiftY][idxCb][idxCr][j][c]	u(1)
}	
}	
}	
}	

F.7.3.2.4 Supplemental enhancement information RBSP syntax

The specifications in clause 7.3.2.4 apply.

F.7.3.2.5 Access unit delimiter RBSP syntax

The specifications in clause 7.3.2.5 apply.

F.7.3.2.6 End of sequence RBSP syntax

The specifications in clause 7.3.2.6 apply.

F.7.3.2.7 End of bitstream RBSP syntax

The specifications in clause 7.3.2.7 apply.

F.7.3.2.8 Filler data RBSP syntax

The specifications in clause 7.3.2.8 apply.

F.7.3.2.9 Slice segment layer RBSP syntax

The specifications in clause 7.3.2.9 apply.

F.7.3.2.10 RBSP slice segment trailing bits syntax

The specifications in clause 7.3.2.10 apply.

F.7.3.2.11 RBSP trailing bits syntax

The specifications in clause 7.3.2.11 apply.

F.7.3.2.12 Byte alignment syntax

The specifications in clause 7.3.2.12 apply.

F.7.3.3 Profile, tier and level syntax

The specifications in clause 7.3.3 apply.

F.7.3.4 Scaling list data syntax

The specifications in clause 7.3.4 apply.

F.7.3.5 Supplemental enhancement information message syntax

The specifications in clause 7.3.5 apply.

F.7.3.6 Slice segment header syntax

F.7.3.6.1 General slice segment header syntax

slice_segment_header( ) {	**Descriptor**
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
if(!dependent_slice_segment_flag) {	
i = 0	
if(num_extra_slice_header_bits > i) {	

i++	
discardable_flag	u(1)
}	
if(num_extra_slice_header_bits > i) {	
i++	
cross_layer_bla_flag	u(1)
}	
for(; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if((nuh_layer_id > 0 && !poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]]) || (nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP))	
slice_pic_order_cnt_lsb	u(v)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
st_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && !default_ref_layers_active_flag && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	

num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumDirectRefLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(slice_type = = P || slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification( )	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_flag && slice_type = = B))	
pred_weight_table( )	
five_minus_max_num_merge_cand	ue(v)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)

slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag || slice_sao_chroma_flag || !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	
if(tiles_enabled_flag || entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
if(poc_reset_info_present_flag)	
poc_reset_idc	u(2)
if(poc_reset_idc != 0)	
poc_reset_period_id	u(6)
if(poc_reset_idc = = 3) {	
full_poc_reset_flag	u(1)
poc_lsb_val	u(v)
}	
if(!PocMsbValRequiredFlag && vps_poc_lsb_aligned_flag)	
poc_msb_cycle_val_present_flag	u(1)
if(poc_msb_cycle_val_present_flag)	
poc_msb_cycle_val	ue(v)
while(more_data_in_slice_segment_header_extension( ))	
slice_segment_header_extension_data_bit	u(1)
}	
byte_alignment( )	
}	

F.7.3.6.2 Reference picture list modification syntax

The specifications in clause 7.3.6.2 apply.

F.7.3.6.3 Weighted prediction parameters syntax

The specifications in clause 7.3.6.3 apply.

F.7.3.7 Short-term reference picture set syntax

The specifications in clause 7.3.7 apply.

F.7.3.8 Slice segment data syntax

F.7.3.8.1 General slice segment data syntax

The specifications in clause 7.3.8.1 apply.

F.7.3.8.2 Coding tree unit syntax

The specifications in clause 7.3.8.2 apply.

F.7.3.8.3 Sample adaptive offset syntax

The specifications in clause 7.3.8.3 apply.

F.7.3.8.4 Coding quadtree syntax

The specifications in clause 7.3.8.4 apply.

F.7.3.8.5 Coding unit syntax

The specifications in clause 7.3.8.5 apply.

F.7.3.8.6 Prediction unit syntax

The specifications in clause 7.3.8.6 apply.

F.7.3.8.7 PCM sample syntax

The specifications in clause 7.3.8.7 apply.

F.7.3.8.8 Transform tree syntax

The specifications in clause 7.3.8.8 apply.

F.7.3.8.9 Motion vector difference syntax

The specifications in clause 7.3.8.9 apply.

F.7.3.8.10 Transform unit syntax

The specifications in clause 7.3.8.10 apply.

F.7.3.8.11 Residual coding syntax

The specifications in clause 7.3.8.11 apply.

F.7.3.8.12 Cross-component prediction syntax

The specifications in clause 7.3.8.12 apply.

F.7.3.8.13 Palette mode syntax

The specifications in clause 7.3.8.13 apply.

F.7.3.8.14 Delta QP syntax

The specifications in clause 7.3.8.14 apply.

F.7.3.8.15 Chroma QP offset syntax

The specifications in clause 7.3.8.15 apply.

F.7.4 Semantics

F.7.4.1 General

F.7.4.2 NAL unit semantics

F.7.4.2.1 General NAL unit semantics

The specifications in clause 7.4.2.1 apply.

F.7.4.2.2 NAL unit header semantics

The specifications in clause 7.4.2.2 apply with the replacement of references to Annex C with references to clause F.13 and with the following additions:

The variable CraOrBlaPicFlag is derived as follows:

$$\text{CraOrBlaPicFlag} = (\ \text{nal\_unit\_type} == \text{BLA\_W\_LP} \ || \ \text{nal\_unit\_type} == \text{BLA\_N\_LP} \ || \ \text{nal\_unit\_type} == \text{BLA\_W\_RADL} \ || \ \text{nal\_unit\_type} == \text{CRA\_NUT}\ ) \quad \text{(F-1)}$$

NOTE 1 – When a picture picA that is a CRA picture and belongs to a layer with nuh_layer_id equal to layerId is present in a bitstream and pictures belonging to the layer with nuh_layer_id equal to layerId and preceding, in decoding order, the picture picA are dropped due to layer down-switching followed by layer up-switching, the RASL pictures associated with the picture picA, if any, may have some reference pictures that may not be available for reference unless one of the following conditions is true:

– The access unit auA containing the picture picA is an IRAP access unit, and the picture with nuh_layer_id equal to SmallestLayerId in the access unit auA, if any, has NoClrasOutputFlag equal to 1.

– The value of HandleCraAsBlaFlag is equal to 1 for the CRA picture picA.

A NAL unit with nal_unit_type equal to EOS_NUT and with a particular nuh_layer_id value shall not be present in an access unit, unless a coded picture with that particular nuh_layer_id value is present in the same access unit.

NOTE 2 – Let indepLayerId be nuh_layer_id value of any independent non-base layer with nuh_layer_id greater than SmallestLayerId. Let firstPic be the first picture, in decoding order, among the pictures with nuh_layer_id equal to indepLayerId or IdPredictedLayer[indepLayerId][i] for any value of i in the range of 0 to NumPredictedLayers[indepLayerId] − 1, inclusive, that follows an IRAP picture with NoClrasOutputFlag equal to 1. firstPic has to be an IRAP picture with nuh_layer_id equal to indepLayerId and with LayerResetFlag equal to 1.

When nal_unit_type is equal to AUD_NUT, the value of nuh_layer_id shall be equal to the minimum of the nuh_layer_id values of all VCL NAL units in the access unit.

F.7.4.2.3 Encapsulation of an SODB within an RBSP (informative)

The specifications in clause 7.4.2.3 apply.

F.7.4.2.4 Order of NAL units and association to coded pictures, access units and coded video sequences

F.7.4.2.4.1 General

The specifications in clause 7.4.2.4.1 apply.

F.7.4.2.4.2 Order of VPS, SPS and PPS RBSPs and their activation

The specifications in clause 7.4.2.4.2 apply with the following additions:

A PPS RBSP includes parameters that can be referred to by the coded slice segment NAL units of one or more coded pictures. Each PPS RBSP is initially considered not active for any layer at the start of the operation of the decoding process. At most one PPS RBSP is considered active for each layer at any given moment during the operation of the decoding process and the activation of any particular PPS RBSP for a particular layer results in the deactivation of the previously-active PPS RBSP for the particular layer (if any).

One PPS RBSP may be the active PPS RBSP for more than one layer. When not explicitly specified, the layer a PPS RBSP is active for is inferred to be the current layer in the context where the active PPS RBSP is referred to.

When a PPS RBSP (with a particular value of pps_pic_parameter_set_id) is not active for a particular layer with nuh_layer_id activatingLayerId and it is referred to by a coded slice segment NAL unit (using a value of slice_pic_parameter_set_id equal to the pps_pic_parameter_set_id value) of the particular layer, it is activated for the particular layer. This PPS RBSP is called the active PPS RBSP for the particular layer until it is deactivated by the activation of another PPS RBSP for the particular layer. A PPS RBSP, with that particular value of pps_pic_parameter_set_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId less than or equal to the TemporalId of the PPS NAL unit or provided through external means, and the PPS NAL unit containing the PPS RBSP shall have nuh_layer_id equal to 0, activatingLayerId, or IdRefLayer[activatingLayerId][i] with any value of i in the range of 0 to NumRefLayers[activatingLayerId] − 1, inclusive.

NOTE 1 – All PPSs, regardless of their values of nuh_layer_id or TemporalId, share the same value space for pps_pic_parameter_set_id. In other words, a PPS with nuh_layer_id equal to X, TemporalId equal to Y and pps_pic_parameter_set_id equal to A would update the previously received PPS that has pps_pic_parameter_set_id equal to A and that has nuh_layer_id not equal to X or TemporalId not equal to Y.

An SPS RBSP includes parameters that can be referred to by one or more PPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each SPS RBSP is initially considered not active for any layer at the start of the operation of the decoding process. At most one SPS RBSP is considered active for each layer at any given moment during the operation of the decoding process and the activation of any particular SPS RBSP for a particular layer results in the deactivation of the previously-active SPS RBSP for that particular layer (if any).

One SPS RBSP may be the active SPS RBSP for more than one layer. When not explicitly specified, the layer an SPS RBSP is active for is inferred to be the current layer in the context where the active SPS RBSP is referred to.

When an SPS RBSP (with a particular value of sps_seq_parameter_set_id) is not already active for a particular non-base layer with nuh_layer_id nuhLayerId and it is referred to by activation of a PPS RBSP (in which pps_seq_parameter_set_id is equal to the sps_seq_parameter_set_id value), or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which active_seq_parameter_set_id[layer_sps_idx[LayerIdxInVps[nuhLayerId]]]] is equal to the sps_seq_parameter_set_id value), it is activated for the particular non-base layer. This SPS RBSP is called the active SPS RBSP for the particular non-base layer until it is deactivated by the activation of another SPS RBSP for the particular non-base layer. An SPS RBSP, with the particular value of sps_seq_parameter_set_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId equal to 0 or provided through external means, and the SPS NAL unit containing the SPS RBSP shall have nuh_layer_id equal to 0, nuhLayerId, or IdRefLayer[nuhLayerId][i] with any value of i in the range of 0 to NumRefLayers[nuhLayerId] − 1, inclusive. An activated SPS RBSP for a particular layer shall remain active for the entire CLVS of that particular layer.

The contents of the hrd_parameters( ) syntax structure shall remain unchanged within a sequence of activated SPS RBSPs, in their activation order, from any activated SPS RBSP until the end of the bitstream or up to but excluding an SPS RBSP that is activated within the next access unit in which at least one of the following conditions is true:

– The access unit includes a picture for each nuh_layer_id value in TargetDecLayerIdList and each picture in the access unit is an IDR picture.

– The access unit includes an IRAP picture with nuh_layer_id equal to SmallestLayerId for which NoClrasOutputFlag is equal to 1.

Any SPS NAL unit with any nuh_layer_id value containing the value of sps_seq_parameter_set_id for the active SPS RBSP for a particular layer shall have the same content as that of the active SPS RBSP for the particular layer unless it follows the access unit auA containing the last coded picture for which the active SPS RBSP for the particular layer is required to be active for the particular layer and precedes the first NAL unit succeeding auA in decoding order that activates an SPS RBSP with the same value of seq_parameter_set_id.

NOTE 2 – All SPSs, regardless of their values of nuh_layer_id, share the same value space for sps_seq_parameter_set_id. In other words, an SPS with nuh_layer_id equal to X and sps_seq_parameter_set_id equal to A would update the previously received SPS with nuh_layer_id not equal to X and sps_seq_parameter_set_id equal to A.

When a VPS RBSP (with a particular value of vps_video_parameter_set_id) is not already active and it is referred to by activation of an SPS RBSP (in which sps_video_parameter_set_id is equal to the vps_video_parameter_set_id value), or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which active_video_parameter_set_id is equal to the vps_video_parameter_set_id value), it is activated. This VPS RBSP is called the active VPS RBSP until it is deactivated by the activation of another VPS RBSP. A VPS RBSP, with that particular value of vps_video_parameter_set_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId equal to 0 or provided through external means, and the VPS NAL unit containing the VPS RBSP shall have nuh_layer_id equal to 0 or SmallestLayerId.

An activated VPS RBSP shall remain active until the end of the bitstream or until it is deactivated by another VPS RBSP in an access unit in which at least one of the following conditions is true:

– The access unit includes a picture for each nuh_layer_id value in TargetDecLayerIdList and each picture in the access unit is an IDR picture.

– The access unit includes an IRAP picture with nuh_layer_id equal to SmallestLayerId for which NoClrasOutputFlag is equal to 1.

For any VPS NAL unit within the CVS with any value of nuh_layer_id and the value of vps_video_parameter_set_id equal to that of the active VPS RBSP for a CVS, the VPS RBSP in the VPS NAL unit shall have the same content as that of the active VPS RBSP for the CVS.

All constraints that are expressed on the relationship between the values of the syntax elements and the values of variables derived from those syntax elements in VPSs, SPSs and PPSs and other syntax elements are expressions of constraints that apply only to the active VPS RBSP, the active SPS RBSP for any particular layer and the active PPS RBSP for any particular layer. If any VPS RBSP, SPS RBSP and PPS RBSP is present that is never activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it was activated by reference in an otherwise conforming bitstream.

During operation of the decoding process for NAL units of a non-base layer, the values of parameters of the active VPS RBSP, the active SPS RBSP for the non-base layer and the active PPS RBSP for the non-base layer are considered in effect. For interpretation of SEI messages applicable to a coded picture of a non-base layer, the values of the active VPS RBSP, the active SPS RBSP for the non-base layer and the active PPS RBSP for the non-base layer for the operation of the decoding process for the VCL NAL units of the coded picture are considered in effect unless otherwise specified in the SEI message semantics.

F.7.4.2.4.3 Order of access units association to CVSs

A bitstream conforming to this Specification consists of one or more CVSs.

A CVS consists of one or more access units. The order of NAL units and coded pictures and their association to access units is described in clause F.7.4.2.4.4.

The first access unit of a CVS is an IRAP access unit with NoRaslOutputFlag equal to 1.

It is a requirement of bitstream conformance that, when present, the next access unit after an access unit that contains an end of sequence NAL unit with nuh_layer_id equal to SmallestLayerId or an end of bitstream NAL unit shall be an IRAP access unit.

F.7.4.2.4.4 Order of NAL units and coded pictures and association to access units

This clause specifies the order of NAL units and coded pictures and their association to access unit for CVSs that contain NAL units with nuh_layer_id greater than 0 that are decoded using the decoding processes specified in Annexes F, G and H.

An access unit consists of one or more coded pictures, each with a distinct value of nuh_layer_id, and zero or more non-VCL NAL units. The association of VCL NAL units to coded pictures is described in clause 7.4.2.4.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

A VCL NAL unit is the first VCL NAL unit of an access unit, when all of the following conditions are true:

– first_slice_segment_in_pic_flag is equal to 1.
– At least one of the following conditions is true:
  – The previous picture in decoding order belongs to a different picture order count (POC) resetting period than the picture containing the VCL NAL unit.
  – PicOrderCntVal derived for the VCL NAL unit differs from the PicOrderCntVal of the previous picture in decoding order.

    NOTE 1 – For any two consecutive access units auA and auB that belong to the same POC resetting period, the PicOrderCntVal of pictures in auA cannot be the same as the PicOrderCntVal of pictures in auB.

    NOTE 2 – Additionally, more conditions, e.g., the following conditions, could but does not need to be used:
    – The nuh_layer_id value of the VCL NAL unit is equal to SmallestLayerId.
    – vps_poc_lsb_aligned_flag is equal to 1 and the slice_pic_order_cnt_lsb value of the VCL NAL unit differs from the slice_pic_order_cnt_lsb value of the previous VCL NAL unit in decoding order.

The first of any of the following NAL units preceding the first VCL NAL unit firstVclNalUnitInAu and succeeding the last VCL NAL unit preceding firstVclNalUnitInAu, if any, specifies the start of a new access unit:

– Access unit delimiter NAL unit (when present).
– VPS NAL unit (when present)
– SPS NAL unit (when present)
– PPS NAL unit (when present)
– Prefix SEI NAL unit (when present)
– NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44 (when present)
– NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 (when present)

When there is none of the above NAL units preceding firstVclNalUnitInAu and succeeding the last VCL NAL unit preceding firstVclNalUnitInAu, if any, firstVclNalUnitInAu starts a new access unit.

A coded picture with nuh_layer_id equal to nuhLayerIdA shall precede, in decoding order, all coded pictures with nuh_layer_id greater than nuhLayerIdA in the same access unit.

The order of the coded pictures and non-VCL NAL units within an access unit shall obey the following constraints:

– When an access unit delimiter NAL unit is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit in any access unit.
– When any VPS NAL units, SPS NAL units, PPS NAL units, prefix SEI NAL units, NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44, or NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 are present, they shall not follow the last VCL NAL unit of the access unit.

– NAL units having nal_unit_type equal to FD_NUT or SUFFIX_SEI_NUT, or in the range of RSV_NVCL45..RSV_NVCL47 or UNSPEC56..UNSPEC63 shall not precede the first VCL NAL unit of the access unit.

– When an end of sequence NAL unit with nuh_layer_id nuhLayerId is present, it shall be the last NAL unit with nuh_layer_id equal to nuhLayerId in the access unit other than an end of bitstream NAL unit (when present).

– When an end of bitstream NAL unit is present, it shall be the last NAL unit in the access unit.

F.7.4.2.4.5 Order of VCL NAL units and association to coded pictures

The specifications in clause 7.4.2.4.5 apply.

F.7.4.3 Raw byte sequence payloads, trailing bits and byte alignment semantics

F.7.4.3.1 Video parameter set RBSP semantics

The specifications in clause 7.4.3.1 apply with the replacement of references to Annex C with references to clause F.13, with the replacement of the semantics of the syntax elements vps_extension_flag and vps_extension_data_flag with that specified below and with the following additions:

Each output operation point is associated with an operation point, a list of nuh_layer_id values of the output layers, in increasing order of nuh_layer_id values, denoted as OptLayerIdList, and the OpTid of the associated operation point. The OpLayerIdList of the operation point associated with an output operation point is also referred to as the OpLayerIdList of the output operation point.

vps_extension_flag equal to 0 specifies that no vps_extension( ) syntax structure is present in the VPS RBSP syntax structure. vps_extension_flag equal to 1 specifies that the vps_extension( ) syntax structure is present in the VPS RBSP syntax structure. When MaxLayersMinus1 is greater than 0, vps_extension_flag shall be equal to 1.

vps_extension_alignment_bit_equal_to_one shall be equal to 1.

vps_extension2_flag equal to 0 specifies that no vps_extension_data_flag syntax elements are present in the VPS RBSP syntax structure. vps_extension2_flag equal to 1 specifies that vps_extension_data_flag syntax elements are present in the VPS RBSP syntax structure. Decoders conforming to a profile specified in Annexes A, G or H shall ignore all data that follow the value 1 for vps_extension2_flag in a VPS RBSP.

vps_extension_data_flag may have any value. Its presence and value do not affect the decoding process for the profiles specified in Annexes A, G or H. Decoders conforming to a profile specified in Annexes A, G or H shall ignore all vps_extension_data_flag syntax elements.

F.7.4.3.1.1 Video parameter set extension semantics

splitting_flag equal to 1 indicates that the dimension_id[i][j] syntax elements are not present and that the binary representation of the nuh_layer_id value in the NAL unit header are split into NumScalabilityTypes segments with lengths, in bits, according to the values of dimension_id_len_minus1[j] and that the values of dimension_id[LayerIdxInVps[nuh_layer_id]][j] are inferred from the NumScalabilityTypes segments. splitting_flag equal to 0 indicates that the syntax elements dimension_id[i][j] are present.

NOTE 1 – When splitting_flag is equal to 1, scalability identifiers of the present scalability dimensions can be derived from the nuh_layer_id syntax element in the NAL unit header by a bit masked copy. The respective bit mask for the i-th present scalability dimension is defined by the value of the dimension_id_len_minus1[i] syntax element and dimBitOffset[i] as specified in the semantics of dimension_id_len_minus1[j].

scalability_mask_flag[i] equal to 1 indicates that dimension_id syntax elements corresponding to the i-th scalability dimension in Table F.1 are present. scalability_mask_flag[i] equal to 0 indicates that dimension_id syntax elements corresponding to the i-th scalability dimension are not present.

Table F.1 – Mapping of ScalabiltyId to scalability dimensions

Scalability mask index	Scalability dimension	ScalabilityId mapping
0	Texture or depth	DepthLayerFlag
1	Multiview	ViewOrderIdx
2	Spatial/quality scalability	DependencyId
3	Auxiliary	AuxId
4-15	Reserved	

dimension_id_len_minus1[j] plus 1 specifies the length, in bits, of the dimension_id[i][j] syntax element.

When splitting_flag is equal to 1, the following applies:

– The variable dimBitOffset[0] is set equal to 0 and for j in the range of 1 to NumScalabilityTypes − 1, inclusive, dimBitOffset[j] is derived as follows:

$$dimBitOffset[\ j\ ] = \sum_{dimIdx=0}^{j-1}\left(dimension\_id\_len\_minus1[\ dimIdx\ ] + 1\right) \qquad \text{(F-2)}$$

– The value of dimension_id_len_minus1[NumScalabilityTypes − 1] is inferred to be equal to 5 − dimBitOffset[NumScalabilityTypes − 1].

– The value of dimBitOffset[NumScalabilityTypes] is set equal to 6.

It is a requirement of bitstream conformance that when NumScalabilityTypes is greater than 0, dimBitOffset[NumScalabilityTypes − 1] shall be less than 6.

vps_nuh_layer_id_present_flag equal to 1 specifies that layer_id_in_nuh[i] for i from 1 to MaxLayersMinus1, inclusive, are present. vps_nuh_layer_id_present_flag equal to 0 specifies that layer_id_in_nuh[i] for i from 1 to MaxLayersMinus1, inclusive, are not present.

layer_id_in_nuh[i] specifies the value of the nuh_layer_id syntax element in VCL NAL units of the i-th layer. When i is greater than 0, layer_id_in_nuh[i] shall be greater than layer_id_in_nuh[i − 1]. For any value of i in the range of 0 to MaxLayersMinus1, inclusive, when not present, the value of layer_id_in_nuh[i] is inferred to be equal to i.

For i from 0 to MaxLayersMinus1, inclusive, the variable LayerIdxInVps[layer_id_in_nuh[i]] is set equal to i.

dimension_id[i][j] specifies the identifier of the j-th present scalability dimension type of the i-th layer. The number of bits used for the representation of dimension_id[i][j] is dimension_id_len_minus1[j] + 1 bits.

Depending on splitting_flag, the following applies:

– If splitting_flag is equal to 1, for i from 0 to MaxLayersMinus1, inclusive, and j from 0 to NumScalabilityTypes − 1, inclusive, dimension_id[i][j] is inferred to be equal to ((layer_id_in_nuh[i] & ((1 << dimBitOffset[j + 1]) − 1)) >> dimBitOffset[j]).

– Otherwise (splitting_flag is equal to 0), for j from 0 to NumScalabilityTypes − 1, inclusive, dimension_id[0][j] is inferred to be equal to 0.

The variable ScalabilityId[i][smIdx] specifying the identifier of the smIdx-th scalability dimension type of the i-th layer, and the variables DepthLayerFlag[lId], ViewOrderIdx[lId], DependencyId[lId], and AuxId[lId] specifying the depth flag, the view order index, the spatial/quality scalability identifier and the auxiliary identifier, respectively, of the layer with nuh_layer_id equal to lId are derived as follows:

```
NumViews = 1
for( i = 0; i <= MaxLayersMinus1; i++ ) {
    lId = layer_id_in_nuh[ i ]
    for( smIdx= 0, j = 0; smIdx < 16; smIdx++ ) {
        if( scalability_mask_flag[ smIdx ] )
            ScalabilityId[ i ][ smIdx ] = dimension_id[ i ][ j++ ]
        else
            ScalabilityId[ i ][ smIdx ] = 0
    }
    DepthLayerFlag[ lId ] = ScalabilityId[ i ][ 0 ]
    ViewOrderIdx[ lId ] = ScalabilityId[ i ][ 1 ]
    DependencyId[ lId ] = ScalabilityId[ i ][ 2 ]
    (F-3)
    AuxId[ lId ] = ScalabilityId[ i ][ 3 ]
    if( i > 0 ) {
        newViewFlag = 1
        for( j = 0; j < i; j++ )
            if( ViewOrderIdx[ lId ] = = ViewOrderIdx[ layer_id_in_nuh[ j ] ] )
                newViewFlag = 0
        NumViews += newViewFlag
    }
}
```

AuxId[lId] equal to 0 specifies the layer with nuh_layer_id equal to lId does not contain auxiliary pictures. AuxId[lId] greater than 0 specifies the type of auxiliary pictures in layer with nuh_layer_id equal to lId as specified in Table F.2.

Table F.2 – Mapping of AuxId to the type of auxiliary pictures

AuxId	Name of AuxId	Type of auxiliary pictures	SEI message describing interpretation of auxiliary pictures
1	AUX_ALPHA	Alpha plane	Alpha channel information
2	AUX_DEPTH	Depth picture	Depth representation information
3..127		Reserved	
128..159		Unspecified	
160..255		Reserved	

NOTE 2 – The interpretation of auxiliary pictures associated with AuxId in the range of 128 to 159, inclusive, is specified through means other than the AuxId value.

AuxId[lId] shall be in the range of 0 to 2, inclusive, or 128 to 159, inclusive, for bitstreams conforming to this version of this Specification. Although the value of AuxId[lId] shall be in the range of 0 to 2, inclusive, or 128 to 159, inclusive, in this version of this Specification, decoders shall allow values of AuxId[lId] in the range of 0 to 255, inclusive.

It is a requirement of bitstream conformance that when AuxId[lId] is equal to AUX_ALPHA or AUX_DEPTH, either of the following applies:

– chroma_format_idc is equal to 0 in the active SPS for the layer with nuh_layer_id equal to lId.

– The value of all decoded chroma samples is equal to $1 << (BitDepth_C - 1)$ in all pictures that have nuh_layer_id equal to lId and for which this VPS RBSP is the active VPS RBSP.

SEI messages may describe the interpretation of auxiliary pictures, including their possible association with one or more primary pictures.

NOTE 3 – Unless constrained by the semantics of the SEI messages specifying the interpretation of auxiliary pictures, it is allowed to have two layers with nuh_layer_id values layerIdA and layerIdB such that AuxId[layerIdA] is equal to AuxId[layerIdB], both being greater than 0 and to have all values of ScalabilityId[LayerIdxInVps[layerIdA]][i] equal to ScalabilityId[LayerIdxInVps[layerIdB]][i] for each value of i in the range of 0 to 15, inclusive. SEI messages specifying the interpretation of auxiliary pictures may specify that a picture with nuh_layer_id equal to layerIdA and a picture with nuh_layer_id equal to layerIdB in the same access unit may both be associated with the same primary picture.

view_id_len specifies the length, in bits, of the view_id_val[i] syntax element.

view_id_val[i] specifies the view identifier of the i-th view specified by the VPS. The length of the view_id_val[i] syntax element is view_id_len bits. When not present, the value of view_id_val[i] is inferred to be equal to 0.

For each layer with nuh_layer_id equal to nuhLayerId, the value ViewId[nuhLayerId] is set equal to view_id_val[ViewOrderIdx[nuhLayerId]].

direct_dependency_flag[i][j] equal to 0 specifies that the layer with index j is not a direct reference layer for the layer with index i. direct_dependency_flag[i][j] equal to 1 specifies that the layer with index j is a direct reference layer for the layer with index i. When direct_dependency_flag[i][j] is not present for i and j in the range of 0 to MaxLayersMinus1, it is inferred to be equal to 0.

The variable DependencyFlag[i][j] is derived as follows:

```
for( i = 0; i  <=  MaxLayersMinus1; i++ )
    for( j = 0; j  <=  MaxLayersMinus1; j++ ) {
        DependencyFlag[ i ][ j ] = direct_dependency_flag[ i ][ j ]
        for( k = 0; k < i; k++ )
            if( direct_dependency_flag[ i ][ k ]  &&  DependencyFlag[ k ][ j ] )       (F-4)
                DependencyFlag[ i ][ j ] = 1
    }
```

The variables NumDirectRefLayers[iNuhLId], IdDirectRefLayer[iNuhLId][d], NumRefLayers[iNuhLId], IdRefLayer[iNuhLId][r], NumPredictedLayers[iNuhLId] and IdPredictedLayer[iNuhLId][p] are derived as follows:

```
for( i = 0; i  <=  MaxLayersMinus1; i++ ) {
    iNuhLId = layer_id_in_nuh[ i ]
    for( j = 0, d = 0, r = 0, p = 0; j  <=  MaxLayersMinus1; j++ ) {
        jNuhLid = layer_id_in_nuh[ j ]
```

```
            if( direct_dependency_flag[ i ][ j ] )
                IdDirectRefLayer[ iNuhLId ][ d++ ] = jNuhLid
            if( DependencyFlag[ i ][ j ] )
                IdRefLayer[ iNuhLId ][ r++ ] = jNuhLid                    (F-5)
            if( DependencyFlag[ j ][ i ] )
                IdPredictedLayer[ iNuhLId ][ p++ ] = jNuhLid
        }
        NumDirectRefLayers[ iNuhLId ] = d
        NumRefLayers[ iNuhLId ] = r
        NumPredictedLayers[ iNuhLId ] = p
    }
```

The variables NumIndependentLayers, NumLayersInTreePartition[i] and TreePartitionLayerIdList[i][j] for i in the range of 0 to NumIndependentLayers − 1, inclusive, and j in the range of 0 to NumLayersInTreePartition[i] − 1, inclusive, are derived as follows:

```
for( i = 0; i <= 63; i++ )
    layerIdInListFlag[ i ] = 0
for( i = 0, k = 0; i <= MaxLayersMinus1; i++ ) {
    iNuhLId = layer_id_in_nuh[ i ]
    if( NumDirectRefLayers[ iNuhLId ] = = 0 ) {
        TreePartitionLayerIdList[ k ][ 0 ] = iNuhLId
        for( j = 0, h = 1; j < NumPredictedLayers[ iNuhLId ]; j++ ) {        (F-6)
            predLId = IdPredictedLayer[ iNuhLId ][ j ]
            if( !layerIdInListFlag[ predLId ] ) {
                TreePartitionLayerIdList[ k ][ h++ ] = predLId
                layerIdInListFlag[ predLId ] = 1
            }
        }
        NumLayersInTreePartition[ k++ ] = h
    }
}
NumIndependentLayers = k
```

It is a requirement of bitstream conformance that AuxId[IdDirectRefLayer[nuhLayerIdA][j]] for any values of nuhLayerIdA and j shall be equal to AuxId[nuhLayerIdA], when AuxId[nuhLayerIdA] is in the range of 0 to 2, inclusive.

NOTE 4 – In other words, no prediction takes place between layers with a different value of AuxId, when AuxId is in the range of 0 to 2, inclusive.

num_add_layer_sets specifies the number of additional layer sets. The value of num_add_layer_sets shall be in the range of 0 to 1 023, inclusive. When vps_base_layer_available_flag is equal to 0, the value of num_add_layer_sets shall be greater than 0. When not present, the value of num_add_layer_sets is inferred to be equal to 0.

The variable NumLayerSets is derived as follows:

```
NumLayerSets = vps_num_layer_sets_minus1 + 1 + num_add_layer_sets          (F-7)
```

When num_add_layer_sets is greater than 0, the variables FirstAddLayerSetIdx and LastAddLayerSetIdx are derived as follows:

```
FirstAddLayerSetIdx = vps_num_layer_sets_minus1 + 1                        (F-8)
LastAddLayerSetIdx = FirstAddLayerSetIdx + num_add_layer_sets − 1
```

highest_layer_idx_plus1[i][j] specifies the values of NumLayersInIdList[vps_num_layer_sets_minus1 + 1 + i] and LayerSetLayerIdList[vps_num_layer_sets_minus1 + 1 + i][layerNum] as follows:

```
layerNum = 0
lsIdx = vps_num_layer_sets_minus1 + 1 + i
for( treeIdx = 1; treeIdx < NumIndependentLayers; treeIdx++ )              (F-9)
    for( layerCnt = 0; layerCnt < highest_layer_idx_plus1[ i ][ treeIdx ]; layerCnt++ )
        LayerSetLayerIdList[ lsIdx ][ layerNum++ ] =
TreePartitionLayerIdList[ treeIdx ][ layerCnt ]
NumLayersInIdList[ lsIdx ] = layerNum
```

The value of highest_layer_idx_plus1[i][j] shall be in the range of 0 to NumLayersInTreePartition[j], inclusive.

The length of the highest_layer_idx_plus1[i][j] syntax element is Ceil(Log2(NumLayersInTreePartition[j] + 1)) bits.

It is a requirement of bitstream conformance that NumLayersInIdList[vps_num_layer_sets_minus1 + 1 + i] shall be greater than 0.

vps_sub_layers_max_minus1_present_flag equal to 1 specifies that the syntax elements sub_layers_vps_max_minus1[i] are present. vps_sub_layers_max_minus1_present_flag equal to 0 specifies that the syntax elements sub_layers_vps_max_minus1[i] are not present.

sub_layers_vps_max_minus1[i] plus 1 specifies, for each value of i in the range of (vps_base_layer_internal_flag ? 0 : 1) to MaxLayersMinus1, inclusive, the maximum number of temporal sub-layers that may be present in the CVS for the layer with nuh_layer_id layerId equal to layer_id_in_nuh[i]. When vps_base_layer_internal_flag is equal to 0, sub_layers_vps_max_minus1[0] constrains the access units for which a decoded picture with nuh_layer_id equal to 0 may be provided by external means as follows: a decoded picture with nuh_layer_id equal to 0 cannot be provided by external means for decoding of an access unit with TemporalId greater than sub_layers_vps_max_minus1[0]. The value of sub_layers_vps_max_minus1[i] shall be in the range of 0 to vps_max_sub_layers_minus1, inclusive. When not present, the value of sub_layers_vps_max_minus1[i] is inferred to be equal to vps_max_sub_layers_minus1.

The variable MaxSubLayersInLayerSetMinus1[i] is derived as follows:

```
for( i = 0; i < NumLayerSets; i++ ) {
    maxSlMinus1 = 0
    for( k = 0; k < NumLayersInIdList[ i ]; k++ ) {
        lId = LayerSetLayerIdList[ i ][ k ]                                          (F-10)
        maxSlMinus1 = Max( maxSLMinus1,
sub_layers_vps_max_minus1[ LayerIdxInVps[ lId ] ] )
    }
    MaxSubLayersInLayerSetMinus1[ i ] = maxSlMinus1
}
```

max_tid_ref_present_flag equal to 1 specifies that the syntax element max_tid_il_ref_pics_plus1[i][j] is present. max_tid_ref_present_flag equal to 0 specifies that the syntax element max_tid_il_ref_pics_plus1[i][j] is not present.

max_tid_il_ref_pics_plus1[i][j] equal to 0 specifies that non-IRAP pictures with nuh_layer_id equal to layer_id_in_nuh[i] are not used as source pictures for inter-layer prediction for pictures with nuh_layer_id equal to layer_id_in_nuh[j]. max_tid_il_ref_pics_plus1[i][j] greater than 0 specifies that pictures with nuh_layer_id equal to layer_id_in_nuh[i] and TemporalId greater than max_tid_il_ref_pics_plus1[i][j] − 1 are not used as source pictures for inter-layer prediction for pictures with nuh_layer_id equal to layer_id_in_nuh[j]. When not present, the value of max_tid_il_ref_pics_plus1[i][j] is inferred to be equal to 7.

default_ref_layers_active_flag equal to 1 specifies that for each picture referring to the VPS, the direct reference layer pictures that belong to all direct reference layers of the layer containing the picture and that may be used for inter-layer prediction as specified by the values of sub_layers_vps_max_minus1[i] and max_tid_il_ref_pics_plus1[i][j] are present in the same access unit as the picture and are included in the inter-layer reference picture set of the picture. default_ref_layers_active_flag equal to 0 specifies that the above restriction may or may not apply.

vps_num_profile_tier_level_minus1 plus 1 specifies the number of profile_tier_level( ) syntax structures in the VPS. The value of vps_num_profile_tier_level_minus1 shall be in the range of 0 to 63, inclusive. When vps_max_layers_minus1 is greater than 0, the value of vps_num_profile_tier_level_minus1 shall be greater than or equal to 1.

vps_profile_present_flag[i] equal to 1 specifies that profile and tier information is present in the i-th profile_tier_level( ) syntax structure. vps_profile_present_flag[i] equal to 0 specifies that profile and tier information is not present in the i-th profile_tier_level( ) syntax structure and is inferred as specified in clause F.7.4.4.

num_add_olss specifies the number of OLSs in addition to the first NumLayerSets OLSs specified by the VPS. The value of num_add_olss shall be in the range of 0 to 1 023, inclusive. When not present, the value of num_add_olss is inferred to be equal to 0.

default_output_layer_idc specifies the derivation of the output layers for the OLSs with index in the range of 1 to vps_num_layer_sets_minus1, inclusive. default_output_layer_idc equal to 0 specifies that all layers in each of the OLSs with index in the range of 1 to vps_num_layer_sets_minus1, inclusive, are output layers of their respective OLSs. default_output_layer_idc equal to 1 specifies that only the layer with the highest value of nuh_layer_id such that nuh_layer_id equal to nuhLayerIdA in each of the OLSs with index in the range of 1 to vps_num_layer_sets_minus1, inclusive, is an output layer of its OLS. default_output_layer_idc equal to 2 specifies that the output layers for the OLSs with index in the range of 1 to vps_num_layer_sets_minus1, inclusive, are specified with the syntax elements

output_layer_flag[i][j]. The value of 3 for default_output_layer_idc is reserved for future use by ITU-T | ISO/IEC. Although the value of default_output_layer_idc is required to be less than 3 in this version of this Specification, decoders shall allow a value of default_output_layer_idc equal to 3 to appear in the syntax.

The variable defaultOutputLayerIdc is set equal to Min(default_output_layer_idc, 2).

layer_set_idx_for_ols_minus1[i] plus 1 specifies the index of the layer set for the i-th OLS. The value of layer_set_idx_for_ols_minus1[i] shall be in the range of 0 to NumLayerSets − 2, inclusive. The length of the layer_set_idx_for_ols_minus1[i] syntax element is Ceil(Log2(NumLayerSets − 1)) bits. When not present, the value of layer_set_idx_for_ols_minus1[i] is inferred to be equal to 0.

For i in the range of 0 to NumOutputLayerSets − 1, inclusive, the variable OlsIdxToLsIdx[i] is derived as specified in the following:

```
OlsIdxToLsIdx[ i ] = ( i < NumLayerSets ) ? i : ( layer_set_idx_for_ols_minus1[ i ] + 1 )        (F-11)
```

output_layer_flag[i][j] equal to 1 specifies that the j-th layer in the i-th OLS is an output layer. output_layer_flag[i][j] equal to 0 specifies that the j-th layer in the i-th OLS is not an output layer.

The value of output_layer_flag[0][0] is inferred to be equal to 1.

When defaultOutputLayerIdc is equal to 0 or 1, for i in the range of 0 to vps_num_layer_sets_minus1, inclusive, and j in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] − 1, inclusive, the variable OutputLayerFlag[i][j] is derived as follows:

– If defaultOutputLayerIdc is equal to 0 or LayerSetLayerIdList[OlsIdxToLsIdx[i]][j] is equal to nuhLayerIdA, with nuhLayerIdA being the highest value in LayerSetLayerIdList[OlsIdxToLsIdx[i]], OutputLayerFlag[i][j] is set equal to 1.
– Otherwise, OutputLayerFlag[i][j] is set equal to 0.

For i in the range of (defaultOutputLayerIdc = = 2) ? 0 : (vps_num_layer_sets_minus1 + 1) to NumOutputLayerSets − 1, inclusive, and j in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] − 1, inclusive, the variable OutputLayerFlag[i][j] is set equal to output_layer_flag[i][j].

The variable NumOutputLayersInOutputLayerSet[i] is derived as follows:

```
NumOutputLayersInOutputLayerSet[ i ] = 0
for( j = 0; j < NumLayersInIdList[ OlsIdxToLsIdx[ i ] ]; j++ ) {
   NumOutputLayersInOutputLayerSet[ i ]  +=  OutputLayerFlag[ i ][ j ]                    (F-12)
   if( OutputLayerFlag[ i ][ j ] )
      OlsHighestOutputLayerId[ i ] = LayerSetLayerIdList[ OlsIdxToLsIdx[ i ] ][ j ]
```

It is a requirement of bitstream conformance that NumOutputLayersInOutputLayerSet[i] shall be greater than 0 for i in the range of 0 to NumOutputLayerSets − 1, inclusive.

The variables NumNecessaryLayers[olsIdx] and NecessaryLayerFlag[olsIdx][lIdx] are derived as follows:

```
for( olsIdx = 0; olsIdx < NumOutputLayerSets; olsIdx++ ) {
   lsIdx = OlsIdxToLsIdx[ olsIdx ]
   for( lsLayerIdx = 0; lsLayerIdx < NumLayersInIdList[ lsIdx ]; lsLayerIdx++ )
      NecessaryLayerFlag[ olsIdx ][ lsLayerIdx ] = 0
   for( lsLayerIdx = 0; lsLayerIdx < NumLayersInIdList[ lsIdx ]; lsLayerIdx++ )
      if( OutputLayerFlag[ olsIdx ][ lsLayerIdx ] ) {
         NecessaryLayerFlag[ olsIdx ][ lsLayerIdx ] = 1                                  (F-13)
         currLayerId = LayerSetLayerIdList[ lsIdx ][ lsLayerIdx ]
         for( rLsLayerIdx = 0; rLsLayerIdx < lsLayerIdx; rLsLayerIdx++ ) {
            refLayerId = LayerSetLayerIdList[ lsIdx ][ rLsLayerIdx ]

   if( DependencyFlag[ LayerIdxInVps[ currLayerId ] ][ LayerIdxInVps[ refLayerId ] ] )
               NecessaryLayerFlag[ olsIdx ][ rLsLayerIdx ] = 1
      }
   NumNecessaryLayers[ olsIdx ] = 0
   for( lsLayerIdx = 0; lsLayerIdx < NumLayersInIdList[ lsIdx ]; lsLayerIdx++ )
      NumNecessaryLayers[ olsIdx ]  +=  NecessaryLayerFlag[ olsIdx ][ lsLayerIdx ]
}
```

It is a requirement of bitstream conformance that for each layer index layerIdx in the range of (vps_base_layer_internal_flag ? 0 : 1) to MaxLayersMinus1, inclusive, there shall be at least one OLS with index olsIdx such that NecessaryLayerFlag[olsIdx][lsLayerIdx] is equal to 1 for the value of lsLayerIdx for which LayerSetLayerIdList[OlsIdxToLsIdx[olsIdx]][lsLayerIdx] is equal to layer_id_in_nuh[layerIdx].

NOTE 5 – In other words, each layer has to be an output layer or a reference layer of an output layer in at least one OLS.

It is a requirement of bitstream conformance that when num_add_layer_sets is greater than 0 and olsIdx has any such value that OlsIdxToLsIdx[olsIdx] is in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive, and NumNecessaryLayers[olsIdx] is equal to 1, NecessaryLayerFlag[olsIdx][0] shall be equal to 1.

NOTE 6 – In other words, when an additional layer set has exactly one necessary layer, the necessary layer is required to be the layer with the smallest nuh_layer_id value within the additional layer set.

profile_tier_level_idx[i][j] specifies the index, into the list of profile_tier_level( ) syntax structures in the VPS, of the profile_tier_level( ) syntax structure that applies to the j-th layer of the i-th OLS as specified in clause F.7.4.4. The length of the profile_tier_level_idx[i][j] syntax element is Ceil(Log2(vps_num_profile_tier_level_minus1 + 1)) bits.

When NecessaryLayerFlag[i][j] is equal to 1 and vps_num_profile_tier_level_minus1 is equal to 0, the value of profile_tier_level_idx[i][j] is inferred to be equal to 0.

When vps_base_layer_internal_flag is equal to 1, the following applies:

– If vps_max_layers_minus1 is greater than 0, the value of profile_tier_level_idx[0][0] is inferred to be equal to 1.

– Otherwise (vps_max_layers_minus1 is equal to 0), the value of profile_tier_level_idx[0][0] is inferred to be equal to 0.

When present, the value of profile_tier_level_idx[i][j] shall be in the range of (vps_base_layer_internal_flag ? 0 : 1) to vps_num_profile_tier_level_minus1, inclusive.

alt_output_layer_flag[i] equal to 0 specifies that an alternative output layer is not used for any output layer in the i-th OLS. alt_output_layer_flag[i] equal to 1 specifies that an alternative output layer may be used for the output layer in the i-th OLS.

When not present, the value of alt_output_layer_flag[i] is inferred to be equal to 0.

NOTE 7 – When alt_output_layer_flag[olsIdx] is equal to 0, pictures that do not belong to the output layers of the OLS with index olsIdx are not output. When alt_output_layer_flag[olsIdx] is equal to 1 and a picture belonging to the output layer of the OLS with index olsIdx is not present in an access unit or has PicOutputFlag equal to 0, a picture with the highest nuh_layer_id among those pictures of the access unit for which PicOutputFlag is equal to 1 and having the nuh_layer_id value among the nuh_layer_id values of the reference layers of the output layer is output.

For each value of olsIdx in the range of 0 to NumOutputLayerSets − 1, inclusive, the following applies:

– When alt_output_layer_flag[olsIdx] is equal to 1, the value of pic_output_flag shall be the same in the slice headers of an access unit that have nuh_layer_id value equal to OlsHighestOutputLayerId[olsIdx] or equal to the nuh_layer_id value of any reference layer of the layer with nuh_layer_id equal to OlsHighestOutputLayerId[olsIdx].

– The variable olsBitstream is derived as follows:

  – Let lsIdx be equal to OlsIdxToLsIdx[olsIdx].

  – If lsIdx is less than or equal to vps_num_layer_sets_minus1, let olsBitstream be the output of the sub-bitstream extraction process specified in clause F.10.1 with the following inputs: the current bitstream, tIdTarget equal to 6 and layerIdListTarget equal to LayerSetLayerIdList[lsIdx].

  – Otherwise, let olsBitstream be the output of the sub-bitstream extraction process specified in clause F.10.3 with the following inputs: the current bitstream, tIdTarget equal to 6 and layerIdListTarget equal to LayerSetLayerIdList[lsIdx].

– Let truncatedOlsBitstream be olsBitstream or be formed from the olsBitstream by removing access units preceding, in decoding order, any access unit with an IRAP picture having nuh_layer_id equal to SmallestLayerId.

– It is a requirement of bitstream conformance that when alt_output_layer_flag[olsIdx] is equal to 1, a bitstream that is formed by removing, from the truncatedOlsBitstream, any coded picture that is not used as a reference for prediction for any other picture and is not the only coded picture of an access unit is a conforming bitstream.

NOTE 8 – When alt_output_layer_flag[olsIdx] is equal to 1, encoders are required to set the values of max_vps_dec_pic_buffering_minus1[i][k][j] such that these values suffice also when pictures of an alternative output layer are marked as "needed for output" in the HRD.

vps_num_rep_formats_minus1 plus 1 specifies the number of the following rep_format( ) syntax structures in the VPS. The value of vps_num_rep_formats_minus1 shall be in the range of 0 to 255, inclusive.

rep_format_idx_present_flag equal to 1 specifies that the syntax elements vps_rep_format_idx[i] are present. rep_format_idx_present_flag equal to 0 specifies that the syntax elements vps_rep_format_idx[i] are not present. When not present, the value of rep_format_idx_present_flag is inferred to be equal to 0.

vps_rep_format_idx[i] specifies the index, into the list of rep_format( ) syntax structures in the VPS, of the rep_format( ) syntax structure that applies to the layer with nuh_layer_id equal to layer_id_in_nuh[i]. When not present, the value of vps_rep_format_idx[i] is inferred to be equal to Min(i, vps_num_rep_formats_minus1). The value of vps_rep_format_idx[i] shall be in the range of 0 to vps_num_rep_formats_minus1, inclusive. The number of bits used for the representation of vps_rep_format_idx[i] is Ceil(Log2(vps_num_rep_formats_minus1 + 1)).

max_one_active_ref_layer_flag equal to 1 specifies that at most one picture is used for inter-layer prediction for each picture in the CVS. max_one_active_ref_layer_flag equal to 0 specifies that more than one picture may be used for inter-layer prediction for each picture in the CVS.

vps_poc_lsb_aligned_flag equal to 0 specifies that the value of slice_pic_order_cnt_lsb may or may not be the same in different pictures of an access unit. vps_poc_lsb_aligned_flag equal to 1 specifies that the value of slice_pic_order_cnt_lsb is the same in all pictures of an access unit. Additionally, the value of vps_poc_lsb_aligned_flag affects the decoding process for picture order count in clause F.8.3.1. When not present, the value of vps_poc_lsb_aligned_flag is inferred to be equal to 0.

poc_lsb_not_present_flag[i] equal to 1 specifies that the slice_pic_order_cnt_lsb syntax element is not present in the slice headers of IDR pictures with nuh_layer_id equal to layer_id_in_nuh[i] in the CVS. poc_lsb_not_present_flag[i] equal to 0 specifies that slice_pic_order_cnt_lsb syntax element may or may not be present in the slice headers of IDR pictures with nuh_layer_id equal to layer_id_in_nuh[i] in the CVS. When not present, the value of poc_lsb_not_present_flag[i] is inferred to be equal to 0.

direct_dep_type_len_minus2 plus 2 specifies the number of bits of the direct_dependency_type[i][j] and the direct_dependency_all_layers_type syntax elements. In bitstreams conforming to this version of this Specification the value of direct_dep_type_len_minus2 shall be equal to 0 or 1. Although the value of direct_dep_type_len_minus2 shall be equal to 0 or 1 in this version of this Specification, decoders shall allow other values of direct_dep_type_len_minus2 in the range of 0 to 30, inclusive, to appear in the syntax.

direct_dependency_all_layers_flag equal to 1 specifies that the syntax element direct_dependency_type[i][j] is not present and inferred from direct_dependency_all_layers_type. direct_dependency_all_layers_flag equal to 0 indicates that the syntax element direct_dependency_type[i][j] is present.

direct_dependency_all_layers_type, when present, specifies the inferred value of direct_dependency_type[i][j] for all combinations of i-th and j-th layers. The length of the direct_dependency_all_layers_type syntax element is direct_dep_type_len_minus2 + 2 bits. Although the value of direct_dependency_all_layers_type is required to be in the range of 0 to 6, inclusive, in this version of this Specification, decoders shall allow values of direct_dependency_all_layers_type in the range of 0 to $2^{32} - 2$, inclusive, to appear in the syntax.

direct_dependency_type[i][j] indicates the type of dependency between the layer with nuh_layer_id equal layer_id_in_nuh[i] and the layer with nuh_layer_id equal to layer_id_in_nuh[j]. direct_dependency_type[i][j] equal to 0 specifies that the layer with nuh_layer_id equal to layer_id_in_nuh[j] may be used for inter-layer sample prediction but is not used for inter-layer motion prediction of the layer with nuh_layer_id equal layer_id_in_nuh[i]. direct_dependency_type[i][j] equal to 1 specifies that the layer with nuh_layer_id equal to layer_id_in_nuh[j] may be used for inter-layer motion prediction but is not used for inter-layer sample prediction of the layer with nuh_layer_id equal layer_id_in_nuh[i]. direct_dependency_type[i][j] equal to 2 specifies that the layer with nuh_layer_id equal to layer_id_in_nuh[j] may be used for both inter-layer motion prediction and inter-layer sample prediction of the layer with nuh_layer_id equal layer_id_in_nuh[i]. The length of the direct_dependency_type[i][j] syntax element is direct_dep_type_len_minus2 + 2 bits. Although the value of direct_dependency_type[i][j] shall be in the range of 0 to 2, inclusive, when the layer with nuh_layer_id equal to layer_id_in_nuh[i] conforms to a profile specified in Annexes A, G or H, and in the range of 0 to 6, inclusive, when the layer with nuh_layer_id equal to layer_id_in_nuh[i] conforms to a profile specified in Annex I, decoders shall allow values of direct_dependency_type[i][j] in the range of 0 to $2^{32} - 2$, inclusive, to appear in the syntax.

When direct_dependency_all_layers_flag is equal to 1, for any i in the range of (1 − vps_base_layer_internal_flag) + 1 to MaxLayersMinus1, inclusive, and any j in the range of 1 − vps_base_layer_internal_flag to i − 1, inclusive, when direct_dependency_flag[i][j] is equal to 1, the value of direct_dependency_type[i][j] is inferred to be equal to direct_dependency_all_layers_type.

When vps_base_layer_internal_flag is equal to 0, for any i in the range of 1 to MaxLayersMinus1, inclusive, when direct_dependency_flag[i][0] is equal to 1, the value of direct_dependency_type[i][0] is inferred to be equal to 0.

The variables VpsInterLayerSamplePredictionEnabled[i][j] and VpsInterLayerMotionPredictionEnabled[i][j] are derived as follows:

```
if( direct_dependency_flag[ i ][ j ] )
    VpsInterLayerSamplePredictionEnabled[ i ][ j ] = ( direct_dependency_type[ i ][ j ] + 1 ) & 0x1
else
    VpsInterLayerSamplePredictionEnabled[ i ][ j ] = 0                                    (F-14)
if( direct_dependency_flag[ i ][ j ] )
    VpsInterLayerMotionPredictionEnabled[ i ][ j ] = ( ( direct_dependency_type[ i ][ j ] + 1 ) & 0x2 ) ?
1 : 0
else
    VpsInterLayerMotionPredictionEnabled[ i ][ j ] = 0
```

vps_non_vui_extension_length specifies the length of the non-VUI VPS extension data following this syntax element and before vps_vui_present_flag, in bytes. The value of vps_non_vui_extension_length shall be in the range of 0 to 4096, inclusive.

vps_non_vui_extension_data_byte may have any value. Decoders shall ignore the value of vps_non_vui_extension_data_byte. Its value does not affect the decoding process specified in this version of this Specification.

vps_vui_present_flag equal to 1 specifies that the vps_vui( ) syntax structure is present in the VPS. vps_vui_present_flag equal to 0 specifies that the vps_vui( ) syntax structure is not present in the VPS.

vps_vui_alignment_bit_equal_to_one shall be equal to 1.

F.7.4.3.1.2 Representation format semantics

chroma_and_bit_depth_vps_present_flag equal to 1 specifies that the syntax elements chroma_format_vps_idc, bit_depth_vps_luma_minus8 and bit_depth_vps_chroma_minus8 are present and that the syntax element separate_colour_plane_vps_flag may be present in the rep_format( ) structure. chroma_and_bit_depth_vps_present_flag equal to 0 specifies that the syntax elements chroma_format_vps_idc, separate_colour_plane_vps_flag, bit_depth_vps_luma_minus8 and bit_depth_vps_chroma_minus8 are not present and are inferred from the previous rep_format( ) syntax structure in the VPS. The value of chroma_and_bit_depth_vps_present_flag of the first rep_format( ) syntax structure in the VPS shall be equal to 1.

pic_width_vps_in_luma_samples, **pic_height_vps_in_luma_samples**, **chroma_format_vps_idc**, **separate_colour_plane_vps_flag**, **bit_depth_vps_luma_minus8** and **bit_depth_vps_chroma_minus8** are used for inference of the values of the SPS syntax elements pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8 and bit_depth_chroma_minus8, respectively, for each SPS that refers to the VPS. When not present in the i-th rep_format( ) syntax structure in the VPS, the value of each of these syntax elements is inferred to be equal to the value of the corresponding syntax element in the (i − 1)-th rep_format( ) syntax structure in the VPS. pic_width_vps_in_luma_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY. pic_height_vps_in_luma_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY. The value of chroma_format_vps_idc shall be in the range of 0 to 3, inclusive. bit_depth_vps_luma_minus8 shall be in the range of 0 to 8, inclusive. bit_depth_vps_chroma_minus8 shall be in the range of 0 to 8, inclusive.

conformance_window_vps_flag equal to 1 specifies that the syntax elements conf_win_vps_left_offset, conf_win_vps_right_offset, conf_win_vps_top_offset and conf_win_vps_bottom_offset follow next in the rep_format( ) structure. conformance_window_vps_flag equal to 0 specifies that the syntax elements conf_win_vps_left_offset, conf_win_vps_right_offset, conf_win_vps_top_offset and conf_win_vps_bottom_offset are not present.

conf_win_vps_left_offset, **conf_win_vps_right_offset**, **conf_win_vps_top_offset** and **conf_win_vps_bottom_offset** are used for inference of the values of the SPS syntax elements, conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset, respectively, for each SPS that refers to the VPS. When not present, the values of conf_win_vps_left_offset, conf_win_vps_right_offset, conf_win_vps_top_offset and conf_win_vps_bottom_offset are inferred to be equal to 0.

The value of SubWidthC * (conf_win_vps_left_offset + conf_win_vps_right_offset) shall be less than pic_width_vps_in_luma_samples and the value of SubHeightC * (conf_win_vps_top_offset + conf_win_vps_bottom_offset) shall be less than pic_height_vps_in_luma_samples.

F.7.4.3.1.3 DPB size semantics

For the lsIdx-th layer set, the number of sub-DPBs is NumLayersInIdList[lsIdx] and for each layer with a particular value of nuh_layer_id in the layer set, the sub-DPB with index layerIdx is assigned, where LayerSetLayerIdList[lsIdx][layerIdx] is equal to nuh_layer_id.

sub_layer_flag_info_present_flag[i] equal to 1 specifies that sub_layer_dpb_info_present_flag[i][j] is present for j in the range of 1 to MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[i]], inclusive. sub_layer_flag_info_present_flag[i] equal to 0 specifies that, for each value of j greater than 0, sub_layer_dpb_info_present_flag[i][j] is not present.

sub_layer_dpb_info_present_flag[i][j] equal to 1 specifies that max_vps_dec_pic_buffering_minus1[i][k][j] may be present for k in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] − 1, inclusive, for the j-th sub-layer of the i-th OLS, and max_vps_num_reorder_pics[i][j] and max_vps_latency_increase_plus1[i][j] are present for the j-th sub-layer of the i-th OLS. sub_layer_dpb_info_present_flag[i][j] equal to 0 specifies that max_vps_dec_pic_buffering_minus1[i][k][j] for k in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] − 1, inclusive, max_vps_num_reorder_pics[i][j] and max_vps_latency_increase_plus1[i][j] are not present. The value of sub_layer_dpb_info_present_flag[i][0] for any possible value of i is inferred to be equal to 1. When not present, the value of sub_layer_dpb_info_present_flag[i][j] for j greater than 0 and any possible value of i, is inferred to be equal to be equal to 0.

max_vps_dec_pic_buffering_minus1[i][k][j] plus 1, when NecessaryLayerFlag[i][k] is equal to 1, specifies the maximum number of decoded pictures, of the k-th layer in the i-th OLS for the CVS, that need to be stored in the DPB when HighestTid is equal to j.

When NecessaryLayerFlag[i][k] is equal to 1, the following applies for i in the range of 1 to NumOutputLayerSets − 1, inclusive:

– When j is greater than 0, max_vps_dec_pic_buffering_minus1[i][k][j] shall be greater than or equal to max_vps_dec_pic_buffering_minus1[i][k][j − 1].

– When max_vps_dec_pic_buffering_minus1[i][0][0] is not present, it is inferred to be equal to 0.

– When max_vps_dec_pic_buffering_minus1[i][k][j] is not present for j in the range of 1 to MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[i]], inclusive, it is inferred to be equal to max_vps_dec_pic_buffering_minus1[i][k][j − 1].

max_vps_num_reorder_pics[i][j] specifies, when HighestTid is equal to j, the maximum allowed number of access units containing a picture with PicOutputFlag equal to 1 that can precede any access unit auA that contains a picture with PicOutputFlag equal to 1 in the i-th OLS in the CVS in decoding order and follow the access unit auA that contains a picture with PicOutputFlag equal to 1 in output order. When max_vps_num_reorder_pics[i][j] is not present for j in the range of 1 to MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[i]], inclusive, due to sub_layer_dpb_info_present_flag[i][j] being equal to 0, it is inferred to be equal to max_vps_num_reorder_pics[i][j − 1].

max_vps_latency_increase_plus1[i][j] not equal to 0 is used to compute the value of MaxVpsLatencyPictures[i][j], which, when HighestTid is equal to j, specifies the maximum number of access units containing a picture with PicOutputFlag equal to 1 in the i-th OLS that can precede any access unit auA that contains a picture with PicOutputFlag equal to 1 in the CVS in output order and follow the access unit auA that contains a picture with PicOutputFlag equal to 1 in decoding order. When max_vps_latency_increase_plus1[i][j] is not present for j in the range of 1 to MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[i]], inclusive, due to sub_layer_dpb_info_present_flag[i][j] being equal to 0, it is inferred to be equal to max_vps_latency_increase_plus1[i][j − 1].

When max_vps_latency_increase_plus1[i][j] is not equal to 0, the value of MaxVpsLatencyPictures[i][j] is specified as follows:

$$\text{MaxVpsLatencyPictures[i][j]} = \text{max\_vps\_num\_reorder\_pics[i][j]} + \text{max\_vps\_latency\_increase\_plus1[i][j]} - 1 \quad \text{(F-15)}$$

When max_vps_latency_increase_plus1[i][j] is equal to 0, no corresponding limit is expressed. The value of max_vps_latency_increase_plus1[i][j] shall be in the range of 0 to $2^{32} - 2$, inclusive.

F.7.4.3.1.4 VPS VUI semantics

NOTE 1 – When vps_vui_present_flag is equal to 0, some of the VPS VUI flags, such as cross_layer_pic_type_aligned_flag, tiles_not_in_use_flag and wpp_not_in_use_flag, are not present and no value is inferred for them. In this case, the semantics of these flags are unspecified and it should be interpreted as such that it is unknown whether the constraints associated with these flags being equal to 1 apply or not; in other words, in this case those constraints may or may not apply.

cross_layer_pic_type_aligned_flag equal to 1 specifies that within a CVS that refers to the VPS, all VCL NAL units that belong to an access unit have the same value of nal_unit_type. cross_layer_pic_type_aligned_flag equal to 0 specifies that within a CVS that refers to the VPS, all VCL NAL units in each access unit may or may not have the same value of nal_unit_type.

cross_layer_irap_aligned_flag equal to 1 specifies that IRAP pictures in the CVS are cross-layer aligned, i.e., when a picture pictureA of a layer layerA in an access unit is an IRAP picture, each picture pictureB in the same access unit that

belongs to a direct reference layer of layerA or that belongs to a layer for which layerA is a direct reference layer of that layer is an IRAP picture and the VCL NAL units of pictureB have the same value of nal_unit_type as that of pictureA. cross_layer_irap_aligned_flag equal to 0 specifies that the above restriction may or may not apply. When not present, the value of cross_layer_irap_aligned_flag is inferred to be equal to vps_vui_present_flag.

all_layers_idr_aligned_flag equal to 1 indicates that within each access unit for which the VCL NAL units refer to the VPS, when one picture is an IRAP picture, all the pictures in the same access unit are IDR pictures and have the same value of nal_unit_type. all_layers_idr_aligned_flag equal to 0 specifies that the above restriction may or may not apply. When not present, the value of all_layers_idr_aligned_flag is inferred to be equal to 0.

bit_rate_present_vps_flag equal to 1 specifies that the syntax element bit_rate_present_flag[i][j] is present. bit_rate_present_vps_flag equal to 0 specifies that the syntax element bit_rate_present_flag[i][j] is not present.

pic_rate_present_vps_flag equal to 1 specifies that the syntax element pic_rate_present_flag[i][j] is present. pic_rate_present_vps_flag equal to 0 specifies that the syntax element pic_rate_present_flag[i][j] is not present.

bit_rate_present_flag[i][j] equal to 1 specifies that the bit rate information for the j-th subset of the i-th layer set is present. bit_rate_present_flag[i] equal to 0 specifies that the bit rate information for the j-th subset of the i-th layer set is not present. The j-th subset of a layer set is derived as follows:

– If i is less than or equal to vps_num_layer_sets_minus1, the j-th subset of a layer set is the output of the sub-bitstream extraction process specified in clause F.10.1, when it is invoked with inBitstream equal to the layer set, tIdTarget equal to j and layerIdListTarget equal to the layer identifier list associated with the layer set as inputs.

– Otherwise (i is greater than vps_num_layer_sets_minus1), the j-th subset of a layer set is the output of the sub-bitstream extraction process specified in clause F.10.3, when it is invoked with inBitstream equal to the layer set, tIdTarget equal to j and layerIdListTarget equal to the layer identifier list associated with the layer set as inputs.

When not present, the value of bit_rate_present_flag[i][j] is inferred to be equal to 0.

pic_rate_present_flag[i][j] equal to 1 specifies that picture rate information for the j-th subset of the i-th layer set is present. pic_rate_present_flag[i][j] equal to 0 specifies that picture rate information for the j-th subset of the i-th layer set is not present. When not present, the value of pic_rate_present_flag[i][j] is inferred to be equal to 0.

avg_bit_rate[i][j] indicates the average bit rate of the j-th subset of the i-th layer set, in bits per second. The value is given by BitRateBPS(avg_bit_rate[i][j]) with the function BitRateBPS( ) being specified as follows:

$$\text{BitRateBPS}(\ x\ ) = (\ x\ \&\ (\ 2^{14} - 1\ )\ ) * 10^{(\ 2 + (\ x\ >>\ 14\ )\ )} \qquad \text{(F-16)}$$

The average bit rate is derived according to the access unit removal time specified in clause F.13. In the following, bTotal is the number of bits in all NAL units of the j-th subset of the i-th layer set, t_1 is the removal time (in seconds) of the first access unit to which the VPS applies and t_2 is the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies. With x specifying the value of avg_bit_rate[i][j], the following applies:

– If t_1 is not equal to t_2, the following condition shall be true:

$$(\ x\ \&\ (\ 2^{14} - 1\ )\ )\ ==\ \text{Round}(\ \text{bTotal} \div (\ (\ t_2 - t_1\ ) * 10^{(\ 2 + (\ x\ >>\ 14\ )\ )}\ )\ ) \qquad \text{(F-17)}$$

– Otherwise (t_1 is equal to t_2), the following condition shall be true:

$$(\ x\ \&\ (\ 2^{14} - 1\ )\ )\ ==\ 0 \qquad \text{(F-18)}$$

max_bit_rate_layer[i][j] indicates an upper bound for the bit rate of the j-th subset of the i-th layer set in any one-second time window of access unit removal time as specified in clause F.13. The upper bound for the bit rate in bits per second is given by BitRateBPS(max_bit_rate_layer[i][j]). The bit rate values are derived according to the access unit removal time specified in clause F.13. In the following, t_1 is any point in time (in seconds), t_2 is set equal to $t_1 + 1 \div 100$ and bTotal is the number of bits in all NAL units of access units with a removal time greater than or equal to t_1 and less than t_2. With x specifying the value of max_bit_rate_layer[i][j], the following condition shall be obeyed for all values of t_1:

$$(\ x\ \&\ (\ 2^{14} - 1\ )\ )\ >=\ \text{bTotal} \div (\ (\ t_2 - t_1\ ) * 10^{(\ 2 + (\ x\ >>\ 14\ )\ )}\ ) \qquad \text{(F-19)}$$

constant_pic_rate_idc[i][j] indicates whether the picture rate of the j-th subset of the i-th layer set is constant. In the following, a temporal segment tSeg is any set of two or more consecutive access units, in decoding order, of the j-th subset of the i-th layer set, auTotal(tSeg) is the number of access units in the temporal segment tSeg, t_1(tSeg) is the removal time (in seconds) of the first access unit (in decoding order) of the temporal segment tSeg, t_2(tSeg) is the removal time (in seconds) of the last access unit (in decoding order) of the temporal segment tSeg, and avgPicRate(tSeg) is the average picture rate in the temporal segment tSeg, and is specified as follows:

$$\text{avgPicRate}(\ \text{tSeg}\ ) = \text{Round}(\ \text{auTotal}(\ \text{tSeg}\ ) * 256 \div (\ t_2(\ \text{tSeg}\ ) - t_1(\ \text{tSeg}\ )\ )\ ) \qquad \text{(F-20)}$$

If the j-th subset of the i-th layer set only contains one or two access units or the value of avgPicRate(tSeg) is constant over all the temporal segments, the picture rate is constant; otherwise, the picture rate is not constant.

constant_pic_rate_idc[i][j] equal to 0 indicates that the picture rate of the j-th subset of the i-th layer set is not constant. constant_pic_rate_idc[i][j] equal to 1 indicates that the picture rate of the j-th subset of the i-th layer set is constant. constant_pic_rate_idc[i][j] equal to 2 indicates that the picture rate of the j-th subset of the i-th layer set may or may not be constant. The value of constant_pic_rate_idc[i][j] shall be in the range of 0 to 2, inclusive.

avg_pic_rate[i] indicates the average picture rate, in units of picture per 256 seconds, of the j-th subset of the layer set. With auTotal being the number of access units in the j-th subset of the i-th layer set, t_1 being the removal time (in seconds) of the first access unit to which the VPS applies, and t_2 being the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies, the following applies:

– If t_1 is not equal to t_2, the following condition shall be true:

$$\text{avg\_pic\_rate[i]} == \text{Round(auTotal} * 256 \div (t_2 - t_1)) \qquad \text{(F-21)}$$

– Otherwise (t_1 is equal to t_2), the following condition shall be true:

$$\text{avg\_pic\_rate[i]} == 0 \qquad \text{(F-22)}$$

video_signal_info_idx_present_flag equal to 1 specifies that the syntax elements vps_num_video_signal_info_minus1 is present and the syntax element vps_video_signal_info_idx[i] may be present. video_signal_info_idx_present_flag equal to 0 specifies that the syntax elements vps_num_video_signal_info_minus1 and vps_video_signal_info_idx[i] are not present.

vps_num_video_signal_info_minus1 plus 1 specifies the number of the following video_signal_info( ) syntax structures in the VPS. When not present, the value of vps_num_video_signal_info_minus1 is inferred to be equal to MaxLayersMinus1 − (vps_base_layer_internal_flag ? 0 : 1).

vps_video_signal_info_idx[i] specifies the index, into the list of video_signal_info( ) syntax structures in the VPS, of the video_signal_info( ) syntax structure that applies to the layer with nuh_layer_id equal to layer_id_in_nuh[i]. If video_signal_info_idx_present_flag is equal to 0, the value of vps_video_signal_info_idx[i] for each value of i in the range of (vps_base_layer_internal_flag ? 0 : 1) to MaxLayersMinus1, inclusive, is inferred to be equal to i. Otherwise, when vps_num_video_signal_info_minus1 is equal to 0, the value of vps_video_signal_info_idx[i] for each value of i in the range of vps_base_layer_internal_flag to MaxLayersMinus1, inclusive, is inferred to be equal to 0. The value of vps_video_signal_info_idx[i] for each value of i in the range of (vps_base_layer_internal_flag ? 0 : 1) to MaxLayersMinus1, inclusive, shall be in the range of 0 to vps_num_video_signal_info_minus1, inclusive.

tiles_not_in_use_flag equal to 1 indicates that the value of tiles_enabled_flag is equal to 0 for each PPS that is referred to by at least one picture referring to the VPS. tiles_not_in_use_flag equal to 0 indicates that such a restriction may or may not apply.

tiles_in_use_flag[i] equal to 1 indicates that the value of tiles_enabled_flag is equal to 1 for each PPS that is referred to by at least one picture of the i-th layer specified by the VPS. tiles_in_use_flag[i] equal to 0 indicates that such a restriction may or may not apply. When not present, the value of tiles_in_use_flag[i] is inferred to be equal to 0.

loop_filter_not_across_tiles_flag[i] equal to 1 indicates that the value of loop_filter_across_tiles_enabled_flag is equal to 0 for each PPS that is referred to by at least one picture of the i-th layer specified by the VPS. loop_filter_not_across_tiles_flag[i] equal to 0 indicates that such a restriction may or may not apply. When not present, the value of loop_filter_not_across_tiles_flag[i] is inferred to be equal to 0.

tile_boundaries_aligned_flag[i][j] equal to 1 indicates that, when any two samples of one picture of the i-th layer specified by the VPS belong to one tile, the two collocated samples, when both present in the picture of the j-th direct reference layer of the i-th layer, belong to one tile, and when any two samples of one picture of the i-th layer belong to different tiles, the two collocated samples, when both present in the picture of the j-th direct reference layer of the i-th layer belong to different tiles. tile_boundaries_aligned_flag equal to 0 indicates that such a restriction may or may not apply. When not present, the value of tile_boundaries_aligned_flag[i][j] is inferred to be equal to 0.

wpp_not_in_use_flag equal to 1 indicates that the value of entropy_coding_sync_enabled_flag is equal to 0 for each PPS that is referred to by at least one picture referring to the VPS. wpp_not_in_use_flag equal to 0 indicates that such a restriction may or may not apply.

wpp_in_use_flag[i] equal to 1 indicates that the value of entropy_coding_sync_enabled_flag is equal to 1 for each PPS that is referred to by at least one picture of the i-th layer specified by the VPS. wpp_in_use_flag[i] equal to 0 indicates that such a restriction may or may not apply. When not present, the value of wpp_in_use_flag[i] is inferred to be equal to 0.

single_layer_for_non_irap_flag equal to 1 indicates the following:

- If vps_base_layer_internal_flag is equal to 1, single_layer_for_non_irap_flag equal to 1 indicates that either one of the following is true for each access unit for which this VPS is the active VPS:
  - All the VCL NAL units of an access unit have the same nuh_layer_id value.
  - Two nuh_layer_id values are used by the VCL NAL units of an access unit and the picture with the greater nuh_layer_id value is an IRAP picture.
- Otherwise (vps_base_layer_internal_flag is equal to 0), single_layer_for_non_irap_flag equal to 1 indicates that any one of the following is true for each access unit for which this VPS is the active VPS:
  - The decoded picture with nuh_layer_id equal to 0 is not provided for the access unit by external means and the access unit contains one coded picture.
  - The decoded picture with nuh_layer_id equal to 0 is not provided for the access unit by external means, the access unit contains two coded pictures and the picture with the greater nuh_layer_id value is an IRAP picture.
  - The decoded picture with nuh_layer_id equal to 0 is provided for an access unit by external means and the access unit contains one coded picture that is an IRAP picture.

single_layer_for_non_irap_flag equal to 0 indicates that the above constraints may or may not apply. When not present, the value of single_layer_for_non_irap_flag is inferred to be equal to 0.

higher_layer_irap_skip_flag equal to 1 indicates that each IRAP picture currIrapPic is constrained as specified below. currIrapPic is derived as follows for each access unit currAu for which this VPS is the active VPS:

- If vps_base_layer_internal_flag is equal to 1, currAu contains two coded pictures and the picture with the greater nuh_layer_id value is an IRAP picture, let currIrapPic be that IRAP picture.
- Otherwise, if vps_base_layer_internal_flag is equal to 0, a decoded picture with nuh_layer_id equal to 0 is not provided for currAu by external means, currAu contains two coded pictures and the picture with the greater nuh_layer_id value is an IRAP picture, let currIrapPic be that IRAP picture.
- Otherwise, if vps_base_layer_internal_flag is equal to 0, the decoded picture with nuh_layer_id equal to 0 is provided for currAu by external means and the access unit contains one coded picture that is an IRAP picture, let currIrapPic be that IRAP picture.
- Otherwise, currIrapPic is not derived for currAu.

The following constraints apply for each picture currIrapPic:

- For all slices of the IRAP picture:
  - slice_type shall be equal to P.
  - slice_sao_luma_flag and slice_sao_chroma_flag shall both be equal to 0.
  - five_minus_max_num_merge_cand shall be equal to 4.
  - weighted_pred_flag shall be equal to 0 in the PPS that is referred to by the slices.
- For all coding units of the IRAP picture:
  - cu_skip_flag[i][j] shall be equal to 1.

When single_layer_for_non_irap_flag is equal to 0, higher_layer_irap_skip_flag shall be equal to 0. When higher_layer_irap_skip_flag is not present it is inferred to be equal to 0.

NOTE 2 – When vps_base_layer_internal_flag is equal to 1, an encoder may set single_layer_for_non_irap_flag equal to 1 as an indication to a decoder that at most two pictures are present in any access unit and whenever there are two pictures in the same access unit, the one with the higher value of nuh_layer_id is an IRAP picture. The encoder may additionally set higher_layer_irap_skip_flag equal to 1 as an indication to a decoder that whenever there are two pictures in the same access unit, the one with the higher value of nuh_layer_id is an IRAP picture for which the decoded samples can be derived by applying the inter-layer reference picture derivation process specified in clause H.8.1.4 with the other picture with the lower value of nuh_layer_id as input.

ilp_restricted_ref_layers_flag equal to 1 indicates that additional restrictions on inter-layer prediction as specified below apply for each direct reference layer of each layer specified by the VPS. ilp_restricted_ref_layers_flag equal to 0 indicates that additional restrictions on inter-layer prediction may or may not apply. When not present, the value of ilp_restricted_ref_layers_flag is inferred to be equal to 0.

min_spatial_segment_offset_plus1[i][j] indicates the spatial region, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer, by itself or together with min_horizontal_ctu_offset_plus1[i][j], as specified below.

The variables refLayerPicWidthInCtbsY[i][j] and refLayerPicHeightInCtbsY[i][j] are set equal to PicWidthInCtbsY and PicHeightInCtbsY, respectively, of the j-th direct reference layer of the i-th layer.

The value of min_spatial_segment_offset_plus1[i][j] shall be in the range of 0 to refLayerPicWidthInCtbsY[i][j] * refLayerPicHeightInCtbsY[i][j], inclusive. When not present, the value of min_spatial_segment_offset_plus1[i][j] is inferred to be equal to 0.

ctu_based_offset_enabled_flag[i][j] equal to 1 specifies that the spatial region, in units of CTUs, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer is indicated by min_spatial_segment_offset_plus1[i][j] and min_horizontal_ctu_offset_plus1[i][j] together. ctu_based_offset_enabled_flag[i][j] equal to 0 specifies that the spatial region, in units of slice segments, tiles or CTU rows, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer is indicated by min_spatial_segment_offset_plus1[i] only. When not present, the value of ctu_based_offset_enabled_flag[i] is inferred to be equal to 0.

min_horizontal_ctu_offset_plus1[i][j], when ctu_based_offset_enabled_flag[i][j] is equal to 1, indicates the spatial region, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer, together with min_spatial_segment_offset_plus1[i][j], as specified below. The value of min_horizontal_ctu_offset_plus1[i][j] shall be in the range of 0 to refLayerPicWidthInCtbsY[i][j], inclusive.

When ctu_based_offset_enabled_flag[i][j] is equal to 1, the variable minHorizontalCtbOffset[i][j] is derived as follows:

$$\text{minHorizontalCtbOffset[i][j]} = (\text{ min\_horizontal\_ctu\_offset\_plus1[i][j]} > 0) \text{ ? } \\ (\text{ min\_horizontal\_ctu\_offset\_plus1[i][j]} - 1) : (\text{ refLayerPicWidthInCtbsY[i][j]} - 1) \quad \text{(F-23)}$$

For any i in the range of 1 to MaxLayersMinus1, inclusive, and any j in the range of 0 to NumDirectRefLayers[layer_id_in_nuh[i]], inclusive, when min_spatial_segment_offset_plus1[i][j] is greater than 0, it is a requirement of bitstream conformance that the following shall apply:

– Let picA be a picture in the i-th layer and let picB be the direct reference layer picture of picA with nuh_layer_id equal to IdDirectRefLayer[layer_id_in_nuh[i]][j]. The variable colCtbAddr[i][j] that denotes the raster scan address of the collocated CTU, in the direct reference layer picture picB, of the CTU with raster scan address equal to ctbAddr[i] in picA is derived as specified in the following:

  – The variables curPicWidthY, curPicHeightY, curCtbLog2SizeY and curPicWidthInCtbsY are set equal to pic_width_in_luma_samples, pic_height_in_luma_samples, CtbLog2SizeY and PicWidthInCtbsY, respectively, of the i-th layer picture picA.

  – The variables refPicWidthY, refPicHeightY, refCtbLog2SizeY and refPicWidthInCtbsY are set equal to pic_width_in_luma_samples, pic_height_in_luma_samples, CtbLog2SizeY and PicWidthInCtbsY, respectively, of the direct reference layer picture picB.

  – If scaled_ref_layer_offset_present_flag[IdDirectRefLayer[layer_id_in_nuh[i]][j]] in the PPS referred to by the i-th layer picture picA is equal to 1, the variables scaledLeftOffset, scaledTopOffset, scaledRightOffset and scaledBottomOffset are set equal to the left scaled reference layer offset ScaledRefLayerLeftOffset, the top scaled reference layer offset ScaledRefLayerTopOffset, the right scaled reference layer offset ScaledRefLayerRightOffset and the bottom scaled reference layer offset ScaledRefLayerBottomOffset, respectively, for the direct reference layer picture picB of the i-th layer picture picA.

  – Otherwise (scaled_ref_layer_offset_present_flag[IdDirectRefLayer[layer_id_in_nuh[i]][j]] is equal to 0), the variables scaledLeftOffset, scaledTopOffset, scaledRightOffset and scaledBottomOffset are set equal to 0.

  – If ref_region_offset_present_flag[IdDirectRefLayer[layer_id_in_nuh[i]][j]] in the PPS referred to by the i-th layer picture picA is equal to 1, the variables refRegionLeftOffset, refRegionTopOffset, refRegionRightOffset and refRegionBottomOffset are set equal to the left reference region offset RefLayerRegionLeftOffset, the top reference region offset RefLayerRegionTopOffset, the right reference region offset RefLayerRegionRightOffset and the bottom reference region offset RefLayerRegionBottomOffset, respectively, for the direct reference layer picture picB of the i-th layer picture picA.

  – Otherwise (ref_region_offset_present_flag[IdDirectRefLayer[layer_id_in_nuh[i]][j]] is equal to 0), the variables refRegionLeftOffset, refRegionTopOffset, refRegionRightOffset and refRegionBottomOffset are set equal to 0.

  – The variables (xP, yP) specifying the location of the top-left luma sample of the CTU with raster scan address equal to ctbAddr relative to top-left luma sample in picA are derived as follows:

$$\text{xP} = (\text{ ctbAddr[i] \% curPicWidthInCtbsY }) \ll \text{ curCtbLog2SizeY} \quad \text{(F-24)}$$

$$\text{yP} = (\text{ ctbAddr[i] / curPicWidthInCtbsY }) \ll \text{ curCtbLog2SizeY} \quad \text{(F-25)}$$

– The variables (xPCol, yPCol) specifying the collocated luma sample location in picB of the luma sample location (xP, yP) in picA are derived as follows:

$$\text{refRegionWidth} = \text{refPicWidthY} - \text{refRegionLeftOffset} - \text{refRegionRightOffset} \tag{F-26}$$

$$\text{refRegionHeight} = \text{refPicHeightY} - \text{refRegionTopOffset} - \text{refRegionBottomOffset} \tag{F-27}$$

$$\text{scaledRefRegionWidth} = \text{curPicHeightY} - \text{scaledLeftOffset} - \text{scaledRightOffset} \tag{F-28}$$

$$\text{scaledRefRegionHeight} = \text{curPicHeightY} - \text{scaledTopOffset} - \text{scaledBottomOffset} \tag{F-29}$$

$$\text{scaleX} = ((\text{refRegionWidth} \ll 16) + (\text{scaledRefRegionWidth} \gg 1)) / \text{scaledRefRegionWidth} \tag{F-30}$$

$$\text{scaleY} = ((\text{refRegionHeight} \ll 16) + (\text{scaledRefRegionHeight} \gg 1)) / \text{scaledRefRegionHeight} \tag{F-31}$$

$$\text{xPCol} = \text{Clip3}(0, (\text{refPicWidthY} - 1), ((((\text{xP} - \text{scaledLeftOffset}) * \text{scaleX} + (1 \ll 15)) \gg 16) + \text{refRegionLeftOffset})) \tag{F-32}$$

$$\text{yPCol} = \text{Clip3}(0, (\text{refPicHeightY} - 1), ((((\text{yP} - \text{scaledTopOffset}) * \text{scaleY} + (1 \ll 15)) \gg 16) + \text{refRegionTopOffset})) \tag{F-33}$$

– The variable colCtbAddr[i][j] is derived as follows:

$$\text{xColCtb}[i][j] = \text{xPCol} \gg \text{refCtbLog2SizeY} \tag{F-34}$$

$$\text{yColCtb}[i][j] = \text{yPCol} \gg \text{refCtbLog2SizeY} \tag{F-35}$$

$$\text{colCtbAddr}[i][j] = \text{xColCtb}[i][j] + (\text{yColCtb}[i][j] * \text{refPicWidthInCtbsY}) \tag{F-36}$$

– If ctu_based_offset_enabled_flag[i][j] is equal to 0, exactly one of the following applies:

  – In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 0 and the following applies:

    – Let slice segment A be any slice segment of a picture of the i-th layer and ctbAddr[i] be the raster scan address of the last CTU in slice segment A. Let slice segment B be the slice segment that belongs to the same access unit as slice segment A, belongs to the j-th direct reference layer of the i-th layer and contains the CTU with raster scan address colCtbAddr[i][j]. Let slice segment C be the slice segment that is in the same picture as slice segment B and follows slice segment B in decoding order, and between slice segment B and slice segment C there are min_spatial_segment_offset_plus1[i] − 1 slice segments in decoding order. When slice segment C is present, the syntax elements of slice segment A are constrained such that no sample or syntax elements values in slice segment C or any slice segment of the same picture following slice segment C in decoding order are used for inter-layer prediction in the decoding process of any samples within slice segment A.

  – In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 0 and the following applies:

    – Let tile A be any tile in any picture picA of the i-th layer and ctbAddr[i] be the raster scan address of the last CTU in tile A. Let tile B be the tile that is in the picture picB belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that contains the CTU with raster scan address colCtbAddr[i][j]. Let tile C be the tile that is also in picB and follows tile B in decoding order, and between tile B and tile C there are min_spatial_segment_offset_plus1[i] − 1 tiles in decoding order. When tile C is present, the syntax elements of tile A are constrained such that no sample or syntax elements values in tile C or any tile of the same picture following tile C in decoding order are used for inter-layer prediction in the decoding process of any samples within tile A.

  – In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1 and the following applies:

    – Let CTU row A be any CTU row in any picture picA of the i-th layer and ctbAddr[i] be the raster scan address of the last CTU in CTU row A. Let CTU row B be the CTU row that is in the picture picB belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that contains the CTU with raster scan address colCtbAddr[i][j]. Let CTU row C be the CTU row that is also in picB and follows CTU row B in decoding order and between CTU row B and CTU row C there are

min_spatial_segment_offset_plus1[i] − 1 CTU rows in decoding order. When CTU row C is present, the syntax elements of CTU row A are constrained such that no sample or syntax elements values in CTU row C or row of the same picture following CTU row C are used for inter-layer prediction in the decoding process of any samples within CTU row A.

- Otherwise (ctu_based_offset_enabled_flag[i][j] is equal to 1), the following applies:
  - The variable refCtbAddr[i][j] is derived as follows:

$$\begin{aligned}&\text{xOffset}[\ i\ ][\ j\ ] = \\ &(\ (\ \text{xColCtb}[\ i\ ][\ j\ ] + \text{minHorizontalCtbOffset}[\ i\ ][\ j\ ]\ ) > (\ \text{refPicWidthInCtbsY}[\ i\ ][\ j\ ] - 1\ )\ )\ ? \\ &(\ \text{refPicWidthInCtbsY}[\ i\ ][\ j\ ] - 1 - \text{xColCtb}[\ i\ ][\ j\ ]\ ) : \text{minHorizontalCtbOffset}[\ i\ ][\ j\ ]\end{aligned} \quad \text{(F-37)}$$

$$\text{yOffset}[\ i\ ][\ j\ ] = (\ \text{min\_spatial\_segment\_offset\_plus1}[\ i\ ][\ j\ ] - 1\ ) * \text{refPicWidthInCtbsY}[\ i\ ][\ j\ ] \quad \text{(F-38)}$$

$$\text{refCtbAddr}[\ i\ ][\ j\ ] = \text{colCtbAddr}[\ i\ ][\ j\ ] + \text{xOffset}[\ i\ ][\ j\ ] + \text{yOffset}[\ i\ ][\ j\ ] \quad \text{(F-39)}$$

  - Let CTU A be any CTU in any picture picA of the i-th layer, and ctbAddr[i] be the raster scan address ctbAddr of CTU A. Let CTU B be a CTU that is in the picture belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that has raster scan address greater than refCtbAddr[i][j]. When CTU B is present, the syntax elements of CTU A are constrained such that no sample or syntax elements values in CTU B are used for inter-layer prediction in the decoding process of any samples within CTU A.

vps_vui_bsp_hrd_present_flag equal to 0 specifies that no bitstream partition HRD parameters are present in the VPS VUI. vps_vui_bsp_hrd_present_flag equal to 1 specifies that bitstream partition HRD parameters are present in the VPS VUI. When vps_timing_info_present_flag is equal to 0, the value of vps_vui_bsp_hrd_present_flag shall be equal to 0.

base_layer_parameter_set_compatibility_flag[i] equal to 1 specifies that the following constraints apply to the layer with nuh_layer_id equal to layer_id_in_nuh[i]. base_layer_parameter_set_compatibility_flag[i] equal to 0 specifies that the following constraints may or may not apply to the layer with nuh_layer_id equal to layer_id_in_nuh[i].

- Each coded slice segment NAL unit with nuh_layer_id value equal to layer_id_in_nuh[i] referring to the VPS shall refer to a PPS with nuh_layer_id value equal to 0.
- Each coded slice segment NAL unit with nuh_layer_id value equal to layer_id_in_nuh[i] referring to the VPS shall refer to a SPS with nuh_layer_id value equal to 0.
- The values of chroma_format_idc, separate_colour_plane_flag, pic_width_in_luma_samples, pic_height_in_luma_samples, bit_depth_luma_minus8, bit_depth_chroma_minus8, conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset, respectively, of the active SPS for the layer with nuh_layer_id equal to layer_id_in_nuh[i] shall be the same as the values of chroma_format_vps_idc, separate_colour_plane_vps_flag, pic_width_vps_in_luma_samples, pic_height_vps_in_luma_samples, bit_depth_vps_luma_minus8, bit_depth_vps_chroma_minus8, conf_win_vps_left_offset, conf_win_vps_right_offset, conf_win_vps_top_offset and conf_win_vps_bottom_offset, respectively, of the vps_rep_format_idx[i]-th rep_format( ) syntax structure in the active VPS.

F.7.4.3.1.5 Video signal info semantics

video_vps_format, **video_full_range_vps_flag**, **colour_primaries_vps**, **transfer_characteristics_vps** and **matrix_coeffs_vps** are used for inference of the values of the SPS VUI syntax elements video_format, video_full_range_flag, colour_primaries, transfer_characteristics, and matrix_coeffs, respectively, for each SPS that refers to the VPS.

For each of these syntax elements, all constraints, if any, that apply to the value of the corresponding SPS VUI syntax element also apply.

F.7.4.3.1.6 VPS VUI bitstream partition HRD parameters semantics

vps_num_add_hrd_params specifies the number of additional hrd_parameters( ) syntax structures present in the VPS. The value of vps_num_add_hrd_params shall be in the range of 0 to 1 024 − vps_num_hrd_parameters, inclusive.

cprms_add_present_flag[i] equal to 1 specifies that the HRD parameters that are common for all sub-layers are present in the i-th hrd_parameters( ) syntax structure. cprms_add_present_flag[i] equal to 0 specifies that the HRD parameters that are common for all sub-layers are not present in the i-th hrd_parameters( ) syntax structure and are derived to be the same as the (i − 1)-th hrd_parameters( ) syntax structure. When vps_num_hrd_parameters is equal to 0, the value of cprms_add_present_flag[0] is inferred to be equal to 1.

num_sub_layer_hrd_minus1[i] plus 1 specifies the number of sub-layers for which HRD parameters are signalled in

the i-th hrd_parameters( ) syntax structure. The value of num_sub_layer_hrd_minus1[i] shall be in the range of 0 to vps_max_sub_layers_minus1, inclusive.

num_signalled_partitioning_schemes[h] specifies the number of signalled partitioning schemes for the h-th output layer set (OLS). The value of num_signalled_partitioning_schemes[h] shall be in the range of 0 to 16, inclusive. When vps_base_layer_internal_flag is equal to 1, the value of num_signalled_partitioning_schemes[0] is inferred to be equal to 0.

num_partitions_in_scheme_minus1[h][j] plus 1 specifies the number of bitstream partitions for the j-th partitioning scheme of the h-th OLS. The value of num_partitions_in_scheme_minus1[h][j] shall be in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[h]] − 1, inclusive. When vps_base_layer_internal_flag is equal to 1, the value of num_partitions_in_scheme_minus1[0][0] is inferred to be equal to 0. The value of num_partitions_in_scheme_minus1[h][0] is inferred to be equal to NumLayersInIdList[h] − 1.

NOTE – The 0-th partitioning scheme for each OLS is inferred to contain the number of bitstream partitions to be equal to the number of layers in the OLS and each bitstream partition contains one layer of the OLS.

layer_included_in_partition_flag[h][j][k][r] equal to 1 specifies that the r-th layer in the h-th OLS is included in the k-th bitstream partition of the j-th partitioning scheme of the h-th OLS. layer_included_in_partition_flag[h][j][k][r] equal to 0 specifies that the r-th layer in the h-th OLS is not included in the k-th bitstream partition of the j-th partitioning scheme of the h-th OLS. When vps_base_layer_internal_flag is equal to 1, the value of layer_included_in_partition_flag[0][0][0][0] is inferred to be equal to 1. The value of layer_included_in_partition_flag[h][0][k][r] for any value of h in the range of 1 to NumOutputLayerSets − 1, inclusive, is inferred to be equal to (k = = r).

It is a requirement of bitstream conformance that the following constraints apply:

– For the j-th partitioning scheme of the h-th OLS, the bitstream partition with index k1 shall not include reference layers of any layers in the bitstream partition with index k2 for any values of k1 and k2 in the range of 0 to num_partitions_in_scheme_minus1[h][j], inclusive, such that k2 is less than k1.
– When vps_base_layer_internal_flag is equal to 0 and layer_included_in_partition_flag[h][j][k][0] is equal to 1 for any value of h in the range of 1 to NumOutputLayerSets − 1, inclusive, any value of j in the range of 0 to num_signalled_partitioning_schemes[h], inclusive, and any value of k in the range of 0 to num_partitions_in_scheme_minus1[h][j], inclusive, the value of layer_included_in_partition_flag[h][j][k][r] for at least one value of r in the range of 1 to NumLayersInIdList[OlsIdxToLsIdx[h]] − 1, inclusive, shall be equal to 1.
– For each partitioning scheme with index j of the h-th OLS and for each layer with layer index r among the layers in the h-th OLS, there exists one and only one value of k in the range of 0 to num_partitions_in_scheme_minus1[h][j], inclusive, such that layer_included_in_partition_flag[h][j][k][r] is equal to 1.

num_bsp_schedules_minus1[h][i][t] plus 1 specifies the number of delivery schedules specified for bitstream partitions of the i-th partitioning scheme of the h-th OLS when HighestTid is equal to t. The value of num_bsp_schedules_minus1[h][i][t] shall be in the range of 0 to 31, inclusive.

The variable BspSchedCnt[h][i][t] is set equal to num_bsp_schedules_minus1[h][i][t] + 1.

bsp_hrd_idx[h][i][t][j][k] specifies the index of the hrd_parameters( ) syntax structure in the VPS for the j-th delivery schedule specified for the k-th bitstream partition of the i-th partitioning scheme for the h-th OLS when HighestTid is equal to t. The length of the bsp_hrd_idx[h][i][t][j][k] syntax element is Ceil(Log2(vps_num_hrd_parameters + vps_num_add_hrd_params)) bits. The value of bsp_hrd_idx[h][i][t][j][k] shall be in the range of 0 to vps_num_hrd_parameters + vps_num_add_hrd_params − 1, inclusive. When vps_num_hrd_parameters + vps_num_add_hrd_params is equal to 1, the value of bsp_hrd_idx[h][i][t][j][k] is inferred to be equal to 0.

bsp_sched_idx[h][i][t][j][k] specifies the index of the delivery schedule within the sub_layer_hrd_parameters(t) syntax structure of the hrd_parameters( ) syntax structure with the index bsp_hrd_idx[h][i][t][j][k], that is used as the j-th delivery schedule specified for the k-th bitstream partition of the i-th partitioning scheme for the h-th OLS when HighestTid is equal to t. The value of bsp_sched_idx[h][i][t][j][k] shall be in the range of 0 to cpb_cnt_minus1[t], inclusive, where cpb_cnt_minus1[t] is found in the hrd_parameters( ) syntax structure corresponding to the index bsp_hrd_idx[h][i][t][j][k].

The following applies for any j greater than 0, any h in the range of 1 to NumOutputLayerSets − 1, inclusive, any i in the range 0 to num_signalled_partitioning_schemes[h], inclusive, any t in the range of 0 to MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[h]], inclusive, and any k in the range of 0 to num_partitions_in_scheme_minus1[h][i], inclusive:

– For x in the range of 0 to 1, inclusive, the following applies:
  – The variable bsIdx is set equal to bsp_sched_idx[h][i][t][j − x][k].

  - The variables bitRateValueMinus1[x], bitRateDuValueMinus1[x], cpbSizeValueMinus1[x] and cpbSizeDuValueMinus1[x] are set equal to the values of the syntax elements bit_rate_value_minus1[bsIdx], bit_rate_du_value_minus1[bsIdx], cpb_size_value_minus1[bsIdx] and cpb_size_du_value_minus1[bsIdx], respectively, found in the sub_layer_hrd_parameters(t) syntax structure within the hrd_parameters( ) structure with index bsp_hrd_idx[h][i][t][j − x][k].

- It is a requirement of bitstream conformance that all of the following conditions shall be true:
  - bitRateValueMinus1[0] is greater than bitRateValueMinus1[1].
  - bitRateDuValueMinus1[0] is greater than bitRateDuValueMinus1[1].
  - cpbSizeValueMinus1[0] is less than or equal to cpbSizeValueMinus1[1].
  - cpbSizeDuValueMinus1[0] is less than or equal to cpbSizeDuValueMinus1[1].

The arrays BpBitRate and BpbSize for bitstream partition buffer (BPB) are derived as follows:

```
for( h = 1; h < NumOutputLayerSets; h++ )
    for( i = 0; i <= num_signalled_partitioning_schemes[ h ]; i++ )
        for( t = 0; t <= MaxSubLayersInLayerSetMinus1[ OlsIdxToLsIdx[ h ] ]; t++ )
            for( j = 0; j <= num_bsp_schedules_minus1[ h ][ i ][ t ]; j++ )
                for( k = 0; k <= num_partitions_in_scheme_minus1[ h ][ i ]; k++ ) {
                    bsIdx = bsp_sched_idx[ h ][ i ][ t ][ j ][ k ]
                    brDu = ( bit_rate_du_value_minus1[ bsIdx ] + 1 ) * 2^( 6 + bit_rate_scale )          (F-40)
                    brPu = ( bit_rate_value_minus1[ bsIdx ] + 1 ) * 2^( 6 + bit_rate_scale )
                    cpbSizeDu = ( cpb_size_du_value_minus1[ bsIdx ] + 1 ) * 2^( 4 + cpb_size_du_scale )
                    cpbSizePu = ( cpb_size_value_minus1[ bsIdx ] + 1 ) * 2^( 4 + cpb_size_scale )
                    BpBitRate[ h ][ i ][ t ][ j ][ k ] = SubPicHrdFlag ? brDu : brPu
                    BpbSize[ h ][ i ][ t ][ j ][ k ] = SubPicHrdFlag? cpbSizeDu : cpbSizePu
                }
```

where the syntax elements bit_rate_scale, cpb_size_du_scale and cpb_size_scale values are found in the hrd_parameters( ) syntax structure corresponding to the index bsp_hrd_idx[h][i][t][j][k], and the syntax elements bit_rate_du_value_minus1[bsIdx], bit_rate_value_minus1[bsIdx], cpb_size_du_value_minus1[bsIdx] and cpb_size_value_minus1[bsIdx] are found in the sub_layer_hrd_parameters(t) syntax structure within that hrd_parameters( ) syntax structure.

F.7.4.3.2 Sequence parameter set RBSP semantics

F.7.4.3.2.1 General sequence parameter set RBSP semantics

The specifications in clause 7.4.3.2.1 apply with the following additions and modifications:

sps_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the SPS. The value of sps_max_sub_layers_minus1 shall be in the range of 0 to 6, inclusive. The value of sps_max_sub_layers_minus1 shall be less than or equal to vps_max_sub_layers_minus1. When not present, the value of sps_max_sub_layers_minus1 is inferred to be equal to (sps_ext_or_max_sub_layers_minus1 = = 7) ? vps_max_sub_layers_minus1 : sps_ext_or_max_sub_layers_minus1.

sps_ext_or_max_sub_layers_minus1 is used to infer the value of sps_max_sub_layers_minus1 and to derive the value of MultiLayerExtSpsFlag.

It is a requirement of bitstream conformance that the following applies:

- If the SPS is referred to by any current picture that belongs to an independent non-base layer, the value of MultiLayerExtSpsFlag derived from the SPS shall be equal to 0.
- Otherwise, the value of MultiLayerExtSpsFlag derived from the SPS shall be equal to 1.

sps_temporal_id_nesting_flag, when sps_max_sub_layers_minus1 is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the SPS. When vps_temporal_id_nesting_flag is equal to 1, sps_temporal_id_nesting_flag shall be equal to 1. When sps_max_sub_layers_minus1 is equal to 0, sps_temporal_id_nesting_flag shall be equal to 1. When not present, the value of sps_temporal_id_nesting_flag is inferred as follows:

- If sps_max_sub_layers_minus1 is greater than 0, the value of sps_temporal_id_nesting_flag is inferred to be equal to vps_temporal_id_nesting_flag.

– Otherwise, the value of sps_temporal_id_nesting_flag is inferred to be equal to 1.

NOTE 1 – The syntax element sps_temporal_id_nesting_flag is used to indicate that temporal up-switching, i.e., switching from decoding up to any TemporalId tIdN to decoding up to any TemporalId tIdM that is greater than tIdN, is always possible in the CVS.

update_rep_format_flag equal to 1 specifies that sps_rep_format_idx is present and that the sps_rep_format_idx-th rep_format( ) syntax structures in the active VPS applies to the layers that refer to this SPS. update_rep_format_flag equal to 0 specifies that sps_rep_format_idx is not present. When the value of vps_num_rep_formats_minus1 in the active VPS is equal to 0, it is a requirement of bitstream conformance that the value of update_rep_format_flag shall be equal to 0. When not present, the value of update_rep_format_flag is inferred to be equal to 0.

sps_rep_format_idx specifies the index, into the list of rep_format( ) syntax structures in the VPS, of the rep_format( ) syntax structure that applies to the layers that refer to this SPS. The value of sps_rep_format_idx shall be in the range of 0 to vps_num_rep_formats_minus1, inclusive.

When a current picture with nuh_layer_id layerIdCurr greater than 0 refers to an SPS and the layer with nuh_layer_id equal to layerIdCurr is not an independent non-base layer, the values of chroma_format_idc, separate_colour_plane_flag, pic_width_in_luma_samples, pic_height_in_luma_samples, bit_depth_luma_minus8, bit_depth_chroma_minus8, conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset are inferred or constrained as follows:

– The variable repFormatIdx is derived as follows:

  – If update_rep_format_flag is equal to 0, the variable repFormatIdx is set equal to vps_rep_format_idx[LayerIdxInVps[layerIdCurr]].

  – Otherwise (update_rep_format_flag is equal to 1), repFormatIdx is set equal to sps_rep_format_idx.

– If MultiLayerExtSpsFlag derived from the active SPS for the layer with nuh_layer_id equal to layerIdCurr is equal to 0, the values of chroma_format_idc, separate_colour_plane_flag, pic_width_in_luma_samples, pic_height_in_luma_samples, bit_depth_luma_minus8, bit_depth_chroma_minus8, conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset, are inferred to be equal to chroma_format_vps_idc, separate_colour_plane_vps_flag, pic_width_vps_in_luma_samples, pic_height_vps_in_luma_samples, bit_depth_vps_luma_minus8, bit_depth_vps_chroma_minus8, conf_win_vps_left_offset, conf_win_vps_right_offset, conf_win_vps_top_offset and conf_win_vps_bottom_offset, respectively, of the repFormatIdx-th rep_format( ) syntax structure in the active VPS and the values of chroma_format_idc, separate_colour_plane_flag, pic_width_in_luma_samples, pic_height_in_luma_samples, bit_depth_luma_minus8, bit_depth_chroma_minus8, conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset signalled in the active SPS for the layer with nuh_layer_id equal to layerIdCurr are ignored.

  NOTE 2 – The values are inferred from the VPS when a layer that is not an independent layer refers to an SPS that is also referred to by the base layer, in which case the SPS has nuh_layer_id equal to 0. For independent layers, the values of these parameters in the active SPS for the base layer apply.

– Otherwise, the values of chroma_format_idc, separate_colour_plane_flag, pic_width_in_luma_samples, pic_height_in_luma_samples, bit_depth_luma_minus8, bit_depth_chroma_minus8, conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset are inferred to be equal to chroma_format_vps_idc, separate_colour_plane_vps_flag, pic_width_vps_in_luma_samples, pic_height_vps_in_luma_samples, bit_depth_vps_luma_minus8, bit_depth_vps_chroma_minus8, conf_win_vps_left_offset, conf_win_vps_right_offset, conf_win_vps_top_offset and conf_win_vps_bottom_offset respectively, of the repFormatIdx-th rep_format( ) syntax structure in the active VPS.

It is a requirement of bitstream conformance that, when present, the value of chroma_format_idc, pic_width_in_luma_samples, pic_height_in_luma_samples, bit_depth_luma_minus8 or bit_depth_chroma_minus8 shall be less than or equal to chroma_format_vps_idc, pic_width_vps_in_luma_samples, pic_height_vps_in_luma_samples, bit_depth_vps_luma_minus8, or bit_depth_vps_chroma_minus8, respectively, of the vps_rep_format_idx[j]-th rep_format( ) syntax structure in the active VPS, where j is equal to LayerIdxInVps[layerIdCurr].

sps_max_dec_pic_buffering_minus1[i] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The value of sps_max_dec_pic_buffering_minus1[i] shall be in the range of 0 to MaxDpbSize − 1, inclusive, where MaxDpbSize is as specified in clause A.4. When i is greater than 0, sps_max_dec_pic_buffering_minus1[i] shall be greater than or equal to sps_max_dec_pic_buffering_minus1[i − 1]. The value of sps_max_dec_pic_buffering_minus1[i] shall be less than or equal to vps_max_dec_pic_buffering_minus1[i] for each value of i. When sps_max_dec_pic_buffering_minus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1 − 1, inclusive, due to sps_sub_layer_ordering_info_present_flag being equal to 0, it is inferred to be equal to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1].

When sps_max_dec_pic_buffering_minus1[i] is not present for i in the range of 0 to sps_max_sub_layers_minus1, inclusive, due to MultiLayerExtSpsFlag being equal to 1, for a layer that refers to the SPS and has nuh_layer_id equal to currLayerId, the value of sps_max_dec_pic_buffering_minus1[i] is inferred to be equal to max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][i] of the active VPS, where layerIdx is equal to the value such that LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx] is equal to currLayerId.

sps_infer_scaling_list_flag equal to 1 specifies that the syntax elements of the scaling list data syntax structure of the SPS are inferred to be equal to those of the SPS that is active for the layer with nuh_layer_id equal to sps_scaling_list_ref_layer_id. sps_infer_scaling_list_flag equal to 0 specifies that the syntax elements of the scaling list data syntax structure are not inferred. When not present, the value of sps_infer_scaling_list_flag is inferred to be 0.

sps_scaling_list_ref_layer_id specifies the value of the nuh_layer_id of the layer for which the active SPS is associated with the same scaling list data as the current SPS. The value of sps_scaling_list_ref_layer_id shall be in the range of 0 to 62, inclusive.

When vps_base_layer_internal_flag is equal to 0, it is a requirement of bitstream conformance that the value of sps_scaling_list_ref_layer_id, when present, shall be greater than 0.

It is a requirement of bitstream conformance that, when an SPS with nuh_layer_id equal to nuhLayerIdA is active for a layer with nuh_layer_id equal to nuhLayerIdB and sps_infer_scaling_list_flag in the SPS is equal to 1, sps_infer_scaling_list_flag shall be equal to 0 for the SPS that is active for the layer with nuh_layer_id equal to sps_scaling_list_ref_layer_id.

It is a requirement of bitstream conformance that, when an SPS with nuh_layer_id equal to nuhLayerIdA is active for a layer with nuh_layer_id equal to nuhLayerIdB and sps_infer_scaling_list_flag in the SPS equal to 1, the layer with nuh_layer_id equal to sps_scaling_list_ref_layer_id shall be a reference layer of the layer with nuh_layer_id equal to nuhLayerIdB.

It is a requirement of bitstream conformance that, when an SPS is active for a layer with sps_infer_scaling_list_flag in the SPS is equal to 1, scaling_list_enabled_flag shall be equal to 1 for the SPS that is active for the layer with nuh_layer_id equal to sps_scaling_list_ref_layer_id.

F.7.4.3.2.2 Sequence parameter set range extension semantics

The specifications in clause 7.4.3.2.2 apply.

F.7.4.3.2.3 Sequence parameter set screen content coding extension semantics

The specifications in clause 7.4.3.2.3 apply.

F.7.4.3.2.4 Sequence parameter set multilayer extension semantics

inter_view_mv_vert_constraint_flag equal to 1 indicates that vertical component of motion vectors used for inter-layer prediction are constrained in the layers for which this SPS RBSP is the active SPS RBSP. When inter_view_mv_vert_constraint_flag is equal to 1, the vertical component of the motion vectors used for inter-layer prediction shall be less than or equal to 56 in units of luma samples. When inter_view_mv_vert_constraint_flag is equal to 0, no constraint on the vertical component of the motion vectors used for inter-layer prediction is signalled by this flag. When not present, the value of inter_view_mv_vert_constraint_flag is inferred to be equal to 0.

F.7.4.3.3 Picture parameter set RBSP semantics

F.7.4.3.3.1 General picture parameter set RBSP semantics

The specifications in clause 7.4.3.3.1 apply with the replacement of the semantics of pps_scaling_list_data_present_flag as follows:

pps_scaling_list_data_present_flag equal to 1 specifies that the scaling list data used for the pictures referring to the PPS are derived based on the scaling lists specified by the active SPS and the scaling lists specified by the PPS. pps_scaling_list_data_present_flag equal to 0 specifies that the scaling list data used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS (when pps_infer_scaling_list_flag is equal to 0) or is specified by pps_scaling_list_ref_layer_id (when pps_infer_scaling_list_flag is equal to 1). When scaling_list_enabled_flag is equal to 0, the value of pps_scaling_list_data_present_flag shall be equal to 0. When scaling_list_enabled_flag is equal to 1, sps_scaling_list_data_present_flag is equal to 0, pps_scaling_list_data_present_flag is equal to 0 and pps_infer_scaling_list_flag is equal to 0, the default scaling list data are used to derive the array ScalingFactor as specified in clause 7.4.5.

F.7.4.3.3.2 Picture parameter set range extension semantics

The specifications in clause 7.4.3.3.2 apply.

F.7.4.3.3.3 Picture parameter set screen content coding extension semantics

The specifications in clause 7.4.3.3.3 apply.

F.7.4.3.3.4 Picture parameter set multilayer extension semantics

poc_reset_info_present_flag equal to 0 specifies that the syntax element poc_reset_idc is not present in the slice segment headers of the slices referring to the PPS. poc_reset_info_present_flag equal to 1 specifies that the syntax element poc_reset_idc is present in the slice segment headers of the slices referring to the PPS. When not present, the value of poc_reset_info_present_flag is inferred to be equal to 0.

pps_infer_scaling_list_flag equal to 1 specifies that the syntax elements of the scaling list data syntax structure of the PPS are inferred to be equal to those of the PPS that is active for the layer with nuh_layer_id equal to pps_scaling_list_ref_layer_id. pps_infer_scaling_list_flag equal to 0 specifies that the syntax elements of the scaling list data syntax structure of the PPS are not inferred. When scaling_list_enabled_flag is equal to 0, nuh_layer_id of the layer referring to the PPS is equal to 0 or pps_scaling_list_data_present_flag is equal to 1, pps_infer_scaling_list_flag shall be equal to 0. When not present, the value of pps_infer_scaling_list_flag is inferred to be 0.

pps_scaling_list_ref_layer_id specifies the value of nuh_layer_id of the layer for which the active PPS has the same scaling list data as the current PPS.

The value of pps_scaling_list_ref_layer_id shall be in the range of 0 to 62, inclusive.

When vps_base_layer_internal_flag is equal to 0, it is a requirement of bitstream conformance that pps_scaling_list_ref_layer_id, when present, shall be greater than 0. It is a requirement of bitstream conformance that, when a PPS with nuh_layer_id equal to nuhLayerIdA is active for a layer with nuh_layer_id equal to nuhLayerIdB and pps_infer_scaling_list_flag in the PPS is equal to 1, pps_infer_scaling_list_flag shall be equal to 0 for the PPS that is active for the layer with nuh_layer_id equal to pps_scaling_list_ref_layer_id.

It is a requirement of bitstream conformance that, when a PPS with nuh_layer_id equal to nuhLayerIdA is active for a layer with nuh_layer_id equal to nuhLayerIdB and pps_infer_scaling_list_flag in the PPS equal to 1, the layer with nuh_layer_id equal to pps_scaling_list_ref_layer_id shall be a reference layer of the layer with nuh_layer_id equal to nuhLayerIdB.

It is a requirement of bitstream conformance that, when a PPS is active for a layer with pps_infer_scaling_list_flag in the PPS equal to 1, scaling_list_enabled_flag shall be equal to 1 for the SPS that is active for the layer with nuh_layer_id equal to pps_scaling_list_ref_layer_id.

num_ref_loc_offsets specifies the number of reference layer location offsets that are present in the PPS. The value of num_ref_loc_offsets shall be in the range of 0 to vps_max_layers_minus1, inclusive.

ref_loc_offset_layer_id[i] specifies the nuh_layer_id value for which the i-th reference layer location offset parameters are specified.

NOTE – ref_loc_offset_layer_id[i] need not be among the direct reference layers, for example when the spatial correspondence of an auxiliary picture to its associated primary picture is specified.

The i-th reference layer location offset parameters consist of the i-th scaled reference layer offset parameters, the i-th reference region offset parameters and the i-th resampling phase set parameters.

scaled_ref_layer_offset_present_flag[i] equal to 1 specifies that the i-th scaled reference layer offset parameters are present in the PPS. scaled_ref_layer_offset_present_flag[i] equal to 0 specifies that the i-th scaled reference layer offset parameters are not present in the PPS. When not present, the value of scaled_ref_layer_offset_present_flag[i] is inferred to be equal to 0.

The i-th scaled reference layer offset parameters specify the spatial correspondence of a picture referring to this PPS relative to the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i].

scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the sample in the current picture that is collocated with the top-left luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the top-left luma sample of the current picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the picture that refers to this PPS. The value of scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14}-1$, inclusive. When not present, the value of scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the sample in the current picture that is collocated with the top-left luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the top-left luma sample of the current picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the picture that refers to this PPS. The value of scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14}-1$, inclusive. When not present, the value of scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the sample in the current picture that is collocated with the bottom-right luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the current picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the picture that refers to this PPS. The value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the sample in the current picture that is collocated with the bottom-right luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the current picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the picture that refers to this PPS. The value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

Let currTopLeftSample, currBotRightSample, colRefRegionTopLeftSample and colRefRegionBotRightSample be the top-left luma sample of the current picture, the bottom-right luma sample of the current picture, the sample in the current picture that is collocated with the top-left luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], and the sample in the current picture that is collocated with the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], respectively.

When the value of scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionTopLeftSample is located to the right of currTopLeftSample. When the value of scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located to the left of currTopLeftSample.

When the value of scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionTopLeftSample is located below currTopLeftSample. When the value of scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located above currTopLeftSample.

When the value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionBotRightSample is located to the left of currBotRightSample. When the value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located to the right of currBotRightSample.

When the value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionBotRightSample is located above currBotRightSample. When the value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located below currBotRightSample.

ref_region_offset_present_flag[i] equal to 1 specifies that the i-th reference region offset parameters are present in the PPS. ref_region_offset_present_flag[i] equal to 0 specifies that the i-th reference region offset parameters are not present in the PPS. When not present, the value of ref_region_offset_present_flag[i] is inferred to be equal to 0.

The i-th reference region offset parameters specify the spatial correspondence of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] relative to the same decoded picture.

ref_region_left_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the top-left luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the top-left luma sample of the same decoded picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_region_left_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of ref_region_left_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

ref_region_top_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the top-left luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the top-left luma sample of the same decoded picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_region_top_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of ref_region_top_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

ref_region_right_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the same decoded picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_layer_right_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of ref_region_right_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

ref_region_bottom_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the same decoded picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14}-1$, inclusive. When not present, the value of ref_region_bottom_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

Let refPicTopLeftSample, refPicBotRightSample, refRegionTopLeftSample and refRegionBotRightSample be the top-left luma sample of the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], the bottom-right luma sample of the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], the top-left luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], respectively.

When the value of ref_region_left_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionTopLeftSample is located to the right of refPicTopLeftSample. When the value of ref_region_left_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionTopLeftSample is located to the left of refPicTopLeftSample.

When the value of ref_region_top_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionTopLeftSample is located below refPicTopLeftSample. When the value of ref_region_top_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionTopLeftSample is located above refPicTopLeftSample.

When the value of ref_region_right_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionBotRightSample is located to the left of refPicBotRightSample. When the value of ref_region_right_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionBotRightSample is located to the right of refPicBotRightSample.

When the value of ref_region_bottom_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionBotRightSample is located above refPicBotRightSample. When the value of ref_region_bottom_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionBotRightSample is located below refPicBotRightSample.

resample_phase_set_present_flag[i] equal to 1 specifies that the i-th resampling phase set is present in the PPS. resample_phase_set_present_flag[i] equal to 0 specifies that the i-th resampling phase set is not present in the PPS. When not present, the value of resample_phase_set_present_flag[i] is inferred to be equal to 0.

The i-th resampling phase set specifies the phase offsets used in resampling process of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. When the layer specified by ref_loc_offset_layer_id[i] is not a direct reference layer of the current layer, the values of the syntax elements phase_hor_luma[ref_loc_offset_layer_id[i]], phase_ver_luma[ref_loc_offset_layer_id[i]], phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] and phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] are unspecified and shall be ignored by decoders.

phase_hor_luma[ref_loc_offset_layer_id[i]] specifies the luma phase shift in the horizontal direction used in the resampling process of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_hor_luma[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 31, inclusive. When not present, the value of phase_hor_luma[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

phase_ver_luma[ref_loc_offset_layer_id[i]] specifies the luma phase shift in the vertical direction used in the resampling of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_ver_luma[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 31, inclusive. When not present, the value of phase_ver_luma[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] minus 8 specifies the chroma phase shift in the horizontal direction used in the resampling of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 63, inclusive. When not present, the value of phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred to be equal to 8.

phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] minus 8 specifies the chroma phase shift in the vertical direction used in the resampling process of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 63, inclusive. When not present, the value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred as follows:

– If chroma_format_idc is equal to 3 (4:4:4 chroma format), the value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred to be equal to 8.

– Otherwise, the value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred to be equal to (4 * scaledRefRegHeight + refRegHeight / 2) / refRegHeight + 4, where the value of scaledRefRegHeight is equal to the value of ScaledRefRegionHeightInSamplesY derived for the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] of the picture that refers to this PPS, and the value of refRegHeight is equal to RefLayerRegionHeightInSamplesY derived for the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]] of the picture that refers to this PPS.

colour_mapping_enabled_flag equal to 1 specifies the colour mapping process may be applied to the inter-layer reference pictures for the coded pictures referring to the PPS. colour_mapping_enabled_flag equal to 0 specifies that the colour mapping process is not applied to the inter-layer reference pictures for the coded pictures referring to the PPS. When not present, the value of colour_mapping_enabled_flag is inferred to be equal to 0.

It is a requirement of bitstream conformance that, when pps_pic_parameter_set_id is greater than or equal to 8, colour_mapping_enabled_flag shall be equal to 0.

F.7.4.3.3.5 General colour mapping table semantics

num_cm_ref_layers_minus1 specifies the number of the following cm_ref_layer_id[i] syntax elements. The value of num_cm_ref_layers_minus1 shall be in the range of 0 to 61, inclusive.

cm_ref_layer_id[i] specifies the nuh_layer_id value of the associated direct reference layer picture for which the colour mapping table is specified.

cm_octant_depth specifies the maximal split depth of the colour mapping table. In bitstreams conforming to this version of this Specification, the value of cm_octant_depth shall be in the range of 0 to 1, inclusive. Other values for cm_octant_depth are reserved for future use by ITU-T | ISO/IEC.

The variable OctantNumC is derived as follows:

$$\text{OctantNumC} = 1 \ll \text{cm\_octant\_depth} \tag{F-41}$$

cm_y_part_num_log2 specifies the number of partitions of the smallest colour mapping table octant for the luma component. In bitstreams conforming to this version of this Specification, the value of (cm_y_part_num_log2 + cm_octant_depth) shall be in the range of 0 to 3, inclusive. Other values for (cm_y_part_num_log2 + cm_octant_depth) are reserved for future use by ITU-T | ISO/IEC.

The variables OctantNumY and PartNumY are derived as follows:

$$\text{OctantNumY} = 1 \ll (\text{cm\_octant\_depth} + \text{cm\_y\_part\_num\_log2}) \tag{F-42}$$

$$\text{PartNumY} = 1 \ll \text{cm\_y\_part\_num\_log2} \tag{F-43}$$

luma_bit_depth_cm_input_minus8 specifies the sample bit depth of the input luma component of the colour mapping process. The variable BitDepthCmInputY is derived as follows:

$$\text{BitDepthCmInputY} = 8 + \text{luma\_bit\_depth\_cm\_input\_minus8} \tag{F-44}$$

It is a requirement of bitstream conformance that the value of BitDepthCmInputY shall be equal to the bit depth of the luma component of any reference layer with nuh_layer_id equal to one of the cm_ref_layer_id[i], for all i in the range of 0 to num_cm_ref_layers_minus1, inclusive.

chroma_bit_depth_cm_input_minus8 specifies the sample bit depth of the input chroma components of the colour mapping process. The variable BitDepthCmInputC is derived as follows:

$$\text{BitDepthCmInputC} = 8 + \text{chroma\_bit\_depth\_cm\_input\_minus8} \tag{F-45}$$

It is a requirement of bitstream conformance that the value of BitDepthCmInputC shall be equal to the bit depth of the chroma components of any reference layer with nuh_layer_id equal to one of the cm_ref_layer_id[i], for all i in the range of 0 to num_cm_ref_layers_minus1, inclusive.

luma_bit_depth_cm_output_minus8 specifies the sample bit depth of the output luma component of the colour mapping process. The variable BitDepthCmOutputY is derived as follows:

$$\text{BitDepthCmOutputY} = 8 + \text{luma\_bit\_depth\_cm\_output\_minus8} \tag{F-46}$$

It is a requirement of bitstream conformance that the value of BitDepthCmOutputY shall be greater than or equal to BitDepthCmInputY and less than or equal to the bit depth of the luma components of any picture that refers to this PPS.

chroma_bit_depth_cm_output_minus8 specifies the sample bit depth of the output chroma components of the colour mapping process. The variable BitDepthCmOutputC is derived as follows:

$$\text{BitDepthCmOutputC} = 8 + \text{chroma\_bit\_depth\_cm\_output\_minus8} \tag{F-47}$$

It is a requirement of bitstream conformance that the value of BitDepthCmOutputC shall be greater than or equal to BitDepthCmInputC and less than or equal to the bit depth of the chroma components of any picture that refers to this PPS.

cm_res_quant_bits specifies the number of least significant bits to be added to the colour mapping coefficient residual values res_coeff_q and res_coeff_r.

cm_delta_flc_bits_minus1 specifies the delta value used to derive the number of bits used to code the syntax element res_coeff_r. The variable CMResLSBits is set equal to Max(0, (10 + BitDepthCmInputY − BitDepthCmOutputY − cm_res_quant_bits − (cm_delta_flc_bits_minus1 + 1))).

cm_adapt_threshold_u_delta specifies the partitioning threshold of the Cb component used in colour mapping process. When not present, the value of cm_adapt_threshold_u_delta is inferred to be equal to 0. The value of cm_adapt_threshold_u_delta shall be in the range of $-2^{\text{BitDepthCmInputC} - 1}$ to $2^{\text{BitDepthCmInputC} - 1} - 1$, inclusive.

The variable CMThreshU is derived as follows:

$$\text{CMThreshU} = \text{cm\_adapt\_threshold\_u\_delta} + (1 \ll (\text{BitDepthCmInputC} - 1)) \quad \text{(F-48)}$$

cm_adapt_threshold_v_delta specifies the partitioning threshold of the Cr component used in colour mapping process. When not present, the value of cm_adapt_threshold_v_delta is inferred to be equal to 0. The value of cm_adapt_threshold_v_delta shall be in the range of $-2^{\text{BitDepthCmInputC} - 1}$ to $2^{\text{BitDepthCmInputC} - 1} - 1$, inclusive.

The variable CMThreshV is derived as follows:

$$\text{CMThreshV} = \text{cm\_adapt\_threshold\_v\_delta} + (1 \ll (\text{BitDepthCmInputC} - 1)) \quad \text{(F-49)}$$

F.7.4.3.3.6 Colour mapping octants semantics

split_octant_flag equal to 1 specifies that the current colour mapping octant is further split into eight octants with half length in each of the three dimensions. split_octant_flag equal to 0 specifies that the current colour mapping octant is not further split into eight octants. When not present, the value of split_octant_flag is inferred to be equal to 0.

coded_res_flag[idxShiftY][idxCb][idxCr][j] equal to 1 specifies that the residuals for the j-th colour mapping coefficients of the octant with octant index equal to (idxShiftY, idxCb, idxCr) are present. coded_res_flag[idxShiftY][idxCb][idxCr][j] equal to 0 specifies that the residuals for the j-th colour mapping coefficients of the octant with octant index equal to (idxShiftY, idxCb, idxCr) are not present. When not present, the value of coded_res_flag[idxShiftY][idxCb][idxCr][j] is inferred to be equal to 0.

res_coeff_q[idxShiftY][idxCb][idxCr][j][c] specifies the quotient of the residual for the j-th colour mapping coefficient of the c-th colour component of the octant with octant index equal to (idxShiftY, idxCb, idxCr). When not present, the value of res_coeff_q[idxShiftY][idxCb][idxCr][j][c] is inferred to be equal to 0.

res_coeff_r[idxShiftY][idxCb][idxCr][j][c] specifies the remainder of the residual for the j-th colour mapping coefficient of the c-th colour component of the octant with octant index equal to (idxShiftY, idxCb, idxCr). The number of bits used to code res_coeff_r is equal to CMResLSBits. If CMResLSBits is equal to 0, res_coeff_r is not present. When not present, the value of res_coeff_r[idxShiftY][idxCb][idxCr][j][c] is inferred to be equal to 0.

res_coeff_s[idxShiftY][idxCb][idxCr][j][c] specifies the sign of the residual for the j-th colour mapping coefficient of the c-th colour component of the octant with octant index equal to (idxShiftY, idxCb, idxCr). When not present, the value of res_coeff_s[idxShiftY][idxCb][idxCr][j][c] is inferred to be equal to 0.

The variables cmResCoeff[idxShiftY][idxCb][idxCr][j][c], with idxShiftY in the range of 0 to OctantNumY − 1, inclusive, idxCb and idxCr both in the range of 0 to OctantNumC − 1, inclusive, j in the range of 0 to 3, inclusive, and c in the range of 0 to 2, inclusive, are derived as follows:

```
cmResCoeff[ idxShiftY ][ idxCb ][ idxCr ][ j ][ c ] =
            ( 1 − 2 * res_coeff_s[ idxShiftY ][ idxCb ][ idxCr ][ j ][ c ] ) *
            ( ( ( res_coeff_q[ idxShiftY ][ idxCb ][ idxCr ][ j ][ c ] << CMResLSBits ) +   (F-50)
            res_coeff_r[ idxShiftY ][ idxCb ][ idxCr ][ j ][ c ] ) << cm_res_quant_bits )
```

The colour mapping coefficients LutY[idxY][idxCb][idxCr][j], LutCb[idxY][idxCb][idxCr][j], LutCr[idxY][idxCb][idxCr][j] with idxY in the range of 0 and OctantNumY − 1, inclusive, idxCb in the range of 0 and OctantNumC − 1, inclusive, idxCr in the range of 0 and OctantNumC − 1, inclusive, and j in the range of 0 and 3, inclusive, are derived as follows:

```
if( idxY == 0 ) {
    predY[ idxY ][ idxCb ][ idxCr ][ j ] = ( j == 0 ) ? 1 024 : 0
    predCb[ idxY ][ idxCb ][ idxCr ][ j ] = ( j == 1 ) ? 1 024 : 0
    predCr[ idxY ][ idxCb ][ idxCr ][ j ] = ( j == 2 ) ? 1 024 : 0
} else {
    predY[ idxY ][ idxCb ][ idxCr ][ j ] = LutY[ idxY − 1 ][ idxCb ][ idxCr ][ j ]
    predCb[ idxY ][ idxCb ][ idxCr ][ j ] = LutCb[ idxY − 1 ][ idxCb ][ idxCr ][ j ]
    predCr[ idxY ][ idxCb ][ idxCr ][ j ] = LutCr[ idxY − 1 ][ idxCb ][ idxCr ][ j ]     (F-51)
}
```

LutY[idxY][idxCb][idxCr][j] =
(cmResCoeff[idxY][idxCb][idxCr][j][0]) +
predY[idxY][idxCb][idxCr][j]
LutCb[idxY][idxCb][idxCr][j] =
(cmResCoeff[idxY][idxCb][idxCr][j][1]) +
predCb[idxY][idxCb][idxCr][j]
LutCr[idxY][idxCb][idxCr][j] =
(cmResCoeff[idxY][idxCb][idxCr][j][2]) +
predCr[idxY][idxCb][idxCr][j]

It is a requirement of bitstream conformance that the values of LutY[idxY][idxCb][idxCr][i], LutCb[idxY][idxCb][idxCr][i] and LutCr[idxY][idxCb][idxCr][i] shall be in the range of -2^{11} to $2^{11} - 1$, inclusive.

NOTE – When BitDepthCmInputC is not equal to BitDepthCmInputY, the encoder should select values of LutY[idxY][idxCb][idxCr][1], LutY[idxY][idxCb][idxCr][2], LutCb[idxY][idxCb][idxCr][0] and LutCr[idxY][idxCb][idxCr][0] that compensate for the bit depth difference.

F.7.4.3.4 Supplemental enhancement information RBSP semantics

The specifications in clause 7.4.3.4 apply.

F.7.4.3.5 Access unit delimiter RBSP semantics

The specifications in clause 7.4.3.5 apply.

F.7.4.3.6 End of sequence RBSP semantics

The specifications in clause 7.4.3.6 apply with the following additions:

When included in a NAL unit with nuh_layer_id nuhLayerId, the end of sequence RBSP specifies that the next picture having nuh_layer_id equal to nuhLayerId or IdPredictedLayer[nuhLayerId][i] for any value of i in the range of 0 to NumPredictedLayers[nuhLayerId] − 1, inclusive, following the current access unit in the bitstream in decoding order (if any) is an IRAP picture with NoRaslOutputFlag equal to 1 and with nuh_layer_id equal to nuhLayerId.

F.7.4.3.7 End of bitstream RBSP semantics

The specifications in clause 7.4.3.7 apply.

F.7.4.3.8 Filler data RBSP semantics

The specifications in clause 7.4.3.8 apply.

F.7.4.3.9 Slice segment layer RBSP semantics

The specifications in clause 7.4.3.9 apply.

F.7.4.3.10 RBSP slice segment trailing bits semantics

The specifications in clause 7.4.3.10 apply.

F.7.4.3.11 RBSP trailing bits semantics

The specifications in clause 7.4.3.11 apply.

F.7.4.3.12 Byte alignment semantics

The specifications in clause 7.4.3.12 apply.

F.7.4.4 Profile, tier and level semantics

The specifications in clause 7.4.4 apply with the replacement of each reference to "Annex A" with a reference to "Annex A, clause G.11 or clause H.11" and with the following additions and modifications:

The variable offsetVal is set equal to 0 and the variable VpsProfileTierLevel[i] is derived according to the following ordered steps:

1. VpsProfileTierLevel[0] is set equal to the profile_tier_level( ) syntax structure in the base VPS (i.e., in the VPS but not in the VPS extension syntax structure vps_extension( )) and offsetVal is set equal to 1.

2. When vps_max_layers_minus1 is greater than 0 and vps_base_layer_internal_flag is equal to 1 VpsProfileTierLevel[offsetVal] is set equal to the first profile_tier_level( ) syntax structure in the VPS extension, and offsetVal is incremented by 1.
3. For j in the range of 0 to (vps_num_profile_tier_level_minus1 − (vps_base_layer_internal_flag ? 2 : 1)), inclusive, VpsProfileTierLevel[offsetVal + j] is set equal to the j-th profile_tier_level( ) syntax structure signalled after the syntax element vps_num_profile_tier_level_minus1 in the VPS extension.

If vps_base_layer_internal_flag is equal to 0, all bits in the profile_tier_level( ) syntax structure VpsProfileTierLevel[0] shall be equal to 0 and decoders shall ignore the syntax structure VpsProfileTierLevel[0]. Otherwise, the semantics of the profile_tier_level( ) syntax structure VpsProfileTierLevel[0] are specified by the remaining part of the current clause.

The scope to which the profile_tier_level( ) syntax structure applies is specified as follows:

– If the profile_tier_level( ) syntax structure is included in an active SPS for the base layer or is the profile_tier_level( ) syntax structure VpsProfileTierLevel[0], it applies to the OLS containing all layers in the bitstream but with only the base layer being the output layer.
– Otherwise, if the profile_tier_level( ) syntax structure is included in an active SPS for an independent non-base layer with nuh_layer_id equal to layerId, it applies to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables assignedBaseLayerId equal to layerId and tIdTarget equal to 6.
– Otherwise, if all of the following conditions are true, the profile_tier_level( ) syntax structure VpsProfileTierLevel[profile_tier_level_idx[olsIdx][0]] applies to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables assignedBaseLayerId equal to LayerSetLayerIdList[OlsIdxToLsIdx[olsIdx]][0] and tIdTarget equal to 6:
  – num_add_layer_sets is greater than 0.
  – OlsIdxToLsIdx[olsIdx] is in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive.
  – NumLayersInIdList[OlsIdxToLsIdx[olsIdx]] is equal to 1.
  – The profile_tier_level( ) syntax structure VpsProfileTierLevel[profile_tier_level_idx[olsIdx][0]] indicates a profile specified in Annex A.
  – The value of general_inbld_flag in the profile_tier_level( ) syntax structure VpsProfileTierLevel[profile_tier_level_idx[olsIdx][0]] is equal to 1.
– Otherwise, the profile_tier_level( ) syntax structure provides profile, tier and level to which a layer in an OLS conforms and the profile_tier_level_idx[i][j] syntax element specifies that the profile_tier_level( ) syntax structure VpsProfileTierLevel[profile_tier_level_idx[i][j]] applies to the j-th layer of the i-th OLS.

NOTE 1 – When vps_base_layer_internal_flag is equal to 1 and vps_max_layers_minus1 is greater than 0, the value of profile_tier_level_idx[0][0] is inferred to be equal to 1, such that VpsProfileTierLevel[1] applies to the 0-th OLS, which contains only the base layer that is the only output layer.

When the profile_tier_level( ) syntax structure is VpsProfileTierLevel[k] for k greater than 0 and profilePresentFlag is equal to 0, the syntax elements general_profile_space, general_tier_flag, general_profile_idc, general_profile_compatibility_flag[j], general_progressive_source_flag, general_interlaced_source_flag, general_non_packed_constraint_flag, general_frame_only_constraint_flag, general_max_12bit_constraint_flag, general_max_10bit_constraint_flag, general_max_8bit_constraint_flag, general_max_422chroma_constraint_flag, general_max_420chroma_constraint_flag, general_max_monochrome_constraint_flag, general_intra_constraint_flag, general_one_picture_only_constraint_flag, general_lower_bit_rate_constraint_flag, general_max_14bit_constraint_flag, general_reserved_zero_33bits, general_reserved_zero_34bits, general_reserved_zero_7bits, general_reserved_zero_35bits, general_reserved_zero_43bits, general_inbld_flag and general_reserved_zero_1bit are not present in the profile_tier_level( ) syntax structure and each is inferred to be equal to the value of corresponding syntax element of the profile_tier_level( ) syntax structure VpsProfileTierLevel[k − 1].

When the profile_tier_level( ) syntax structure is VpsProfileTierLevel[k] for k greater than 0 and any of the syntax elements sub_layer_profile_space[i], sub_layer_tier_flag[i], sub_layer_profile_idc[i], sub_layer_profile_compatibility_flag[i][j], sub_layer_progressive_source_flag[i], sub_layer_interlaced_source_flag[i], sub_layer_non_packed_constraint_flag[i], sub_layer_frame_only_constraint_flag[i], sub_layer_max_12bit_constraint_flag[i], sub_layer_max_10bit_constraint_flag[i], sub_layer_max_8bit_constraint_flag[i], sub_layer_max_422chroma_constraint_flag[i], sub_layer_max_420chroma_constraint_flag[i], sub_layer_max_monochrome_constraint_flag[i], sub_layer_intra_constraint_flag[i], sub_layer_one_picture_only_constraint_flag[i], sub_layer_lower_bit_rate_constraint_flag[i], sub_layer_max_14bit_constraint_flag[i], sub_layer_reserved_zero_33bits[i], sub_layer_reserved_zero_34bits[i], sub_layer_reserved_zero_7bits[i], sub_layer_reserved_zero_35bits[i], sub_layer_reserved_zero_43bits[i], sub_layer_inbld_flag[i], sub_layer_reserved_zero_1bit[i] and sub_layer_level_idc[i] is not present for any value of i in the range of 0 to maxNumSubLayersMinus1 − 1, inclusive, in the profile_tier_level( ) syntax structure, the value of the syntax element is inferred to be equal to the value of the corresponding syntax element of the profile_tier_level( ) syntax structure VpsProfileTierLevel[k − 1].

A sequence of pictures picSeq is derived as follows:

– If the profile_tier_level( ) syntax structure is included in an SPS, picSeq consists of the pictures in the CLVS of the layer for which the SPS is the active SPS.
– Otherwise, if the profile_tier_level( ) syntax structure is VpsProfileTierLevel[0], picSeq consists of the pictures with nuh_layer_id equal to 0 in the CVS.
– Otherwise, if vps_max_layers_minus1 is greater than 0, vps_base_layer_internal_flag is equal to 1 and the profile_tier_level( ) syntax structure is VpsProfileTierLevel[1], picSeq consists of the pictures with nuh_layer_id equal to 0 in the CVS.
– Otherwise (the profile_tier_level( ) syntax structure is VpsProfileTierLevel[offsetVal + profile_tier_level_idx[i][j]] for any values of i in the range of 0 to NumOutputLayerSets − 1, inclusive, and j in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] − 1, inclusive, such that NecessaryLayerFlag[i][j] is equal to 1), picSeq consists of the pictures with nuh_layer_id equal to LayerSetLayerIdList[OlsIdxToLsIdx[i]][j] in the CVS.

general_progressive_source_flag and **general_interlaced_source_flag** are interpreted as follows:

– If general_progressive_source_flag is equal to 1 and general interlaced_source_flag is equal to 0, the source scan type of the pictures in the picSeq should be interpreted as progressive only.
– Otherwise, if general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 1, the source scan type of the pictures in the picSeq should be interpreted as interlaced only.
– Otherwise, if general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 0, the source scan type of the pictures in the picSeq should be interpreted as unknown or unspecified.
– Otherwise, general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 1, the source scan type of each picture in the picSeq is indicated at the picture level using the syntax element source_scan_type in a picture timing SEI message or the syntax element ffinfo_source_scan_type in a frame-field information SEI message.

NOTE 2 – Decoders may ignore the values of general_progressive_source_flag and general_interlaced_source_flag for purposes other than determining the value to be inferred for frame_field_info_present_flag when vui_parameters_present_flag is equal to 0, as there are no other decoding process requirements associated with the values of these flags. Moreover, the actual source scan type of the pictures is outside the scope of this Specification and the method by which the encoder selects the values of general_progressive_source_flag and general_interlaced_source_flag is unspecified.

general_non_packed_constraint_flag equal to 1 specifies that there are neither frame packing arrangement SEI messages nor segmented rectangular frame packing arrangement SEI messages present for the pictures of picSeq. general non_packed_constraint_flag equal to 0 indicates that there may or may not be one or more frame packing arrangement SEI messages or segmented rectangular frame packing arrangement SEI messages present for the pictures of picSeq.

NOTE 3 – Decoders may ignore the value of general_non_packed_constraint_flag, as there are no decoding process requirements associated with the presence or interpretation of frame packing arrangement SEI messages or segmented rectangular frame packing arrangement SEI messages.

general_frame_only_constraint_flag equal to 1 specifies that field_seq_flag in the active SPSs for the pictures of picSeq is equal to 0. general_frame_only_constraint_flag equal to 0 indicates that field_seq_flag in the active SPSs for the pictures of picSeq may or may not be equal to 0.

NOTE 4 – Decoders may ignore the value of general_frame_only_constraint_flag, as there are no decoding process requirements associated with the value of field_seq_flag.

NOTE 5 – When general_progressive_source_flag is equal to 1, general_frame_only_constraint_flag may or may not be equal to 1.

It is a requirement of bitstream conformance that when olsIdx is such that alt_output_layer_flag[olsIdx] is equal to 1, general_progressive_source_flag, general_interlaced_source_flag, general_non_packed_constraint_flag and general_frame_only_constraint_flag in VpsProfileTierLevel[offsetVal + profile_tier_level_idx[olsIdx][j]] shall be equal to general_progressive_source_flag, general_interlaced_source_flag, general_non_packed_constraint_flag and general_frame_only_constraint_flag, respectively, in VpsProfileTierLevel[offsetVal + profile_tier_level_idx[olsIdx][k]] for any values of j and k in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[olsIdx]] − 1, inclusive, such that NecessaryLayerFlag[olsIdx][j] is equal to 1 and NecessaryLayerFlag[olsIdx][k] is equal to 1.

F.7.4.5 Scaling list data semantics

The specifications in clause 7.4.5 apply.

F.7.4.6 Supplemental enhancement information message semantics

The specifications in clause 7.4.6 apply.

F.7.4.7 Slice segment header semantics

F.7.4.7.1 General slice segment header semantics

The specifications in clause 7.4.7.1 apply with the replacement of references to Annex C with references to clause F.13 and with the following additions:

When present, the value of the slice segment header syntax elements discardable_flag, cross_layer_bla_flag, inter_layer_pred_enabled_flag, num_inter_layer_ref_pics_minus1, poc_reset_idc, poc_reset_period_id, full_poc_reset_flag, poc_lsb_val, poc_msb_cycle_val_present_flag and poc_msb_cycle_val shall be the same in all slice segment headers of a coded picture. When present, the value of the slice segment header syntax elements inter_layer_pred_layer_idc[i] shall be the same in all slice segment headers of a coded picture for each possible value of i.

When nal_unit_type has a value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e., the picture is an IRAP picture, NumDirectRefLayers[nuh_layer_id] is equal to 0, and pps_curr_pic_ref_enabled_flag is equal to 0, slice_type shall be equal to 2.

When sps_max_dec_pic_buffering_minus1[TemporalId] is equal to 0, NumDirectRefLayers[nuh_layer_id] is equal to 0, and pps_curr_pic_ref_enabled_flag is equal to 0, slice_type shall be equal to 2.

discardable_flag equal to 1 specifies that the coded picture is not used as a reference picture for inter prediction and is not used as a source picture for inter-layer prediction in the decoding process of subsequent pictures in decoding order. discardable_flag equal to 0 specifies that the coded picture may be used as a reference picture for inter prediction and may be used as a source picture for inter-layer prediction in the decoding process of subsequent pictures in decoding order. When nal_unit_type is equal to TRAIL_R, TSA_R, STSA_R, RASL_R or RADL_R, the value of discardable_flag shall be equal to 0. When not present, the value of discardable_flag is inferred to be equal to 0.

NOTE 1 – When a coded picture has discardable_flag equal to 1 in its slice headers, the picture may be ignored by decoders without affecting the decoding result of any other picture, in the same layer or a different layer.

cross_layer_bla_flag equal to 1 affects the derivation of NoClrasOutputFlag and LayerResetFlag as specified in clause F.8.1.3. cross_layer_bla_flag shall be equal to 0 for pictures with nal_unit_type not equal to IDR_W_RADL or IDR_N_LP or with nuh_layer_id nuhLayerId not equal to any value for which NumDirectRefLayers[nuhLayerId] is equal to 0. When not present, the value of cross_layer_bla_flag is inferred to be equal to 0.

When vps_extension_flag is equal to 1, the sum of NumNegativePics[CurrRpsIdx], NumPositivePics[CurrRpsIdx], num_long_term_sps and num_long_term_pics shall be less than or equal to MaxDpbSize − 1, where MaxDpbSize is as specified in clause G.11.2 or H.11.2.

NOTE 2 – The way that MaxDpbSize is specified in clause G.11.2 is the same as in clause H.11.2.

inter_layer_pred_enabled_flag equal to 1 specifies that inter-layer prediction may be used in decoding of the current picture. inter_layer_pred_enabled_flag equal to 0 specifies that inter-layer prediction is not used in decoding of the current picture.

num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the num_inter_layer_ref_pics_minus1 syntax element is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of num_inter_layer_ref_pics_minus1 shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] − 1, inclusive.

The variables numRefLayerPics and refLayerPicIdc[j] are derived as follows:

```
for( i = 0, j = 0; i < NumDirectRefLayers[ nuh_layer_id ]; i++ ) {
    refLayerIdx = LayerIdxInVps[ IdDirectRefLayer[ nuh_layer_id ][ i ] ]
    if( sub_layers_vps_max_minus1[ refLayerIdx ] >= TemporalId && ( TemporalId = = 0 | |     (F-52)
            max_tid_il_ref_pics_plus1[ refLayerIdx ][ LayerIdxInVps[ nuh_layer_id ] ] >
TemporalId ) )
        refLayerPicIdc[ j++ ] = i
}
numRefLayerPics = j
```

The variable NumActiveRefLayerPics is derived as follows:

```
if( nuh_layer_id = = 0 | | numRefLayerPics = = 0 )
    NumActiveRefLayerPics = 0
else if( default_ref_layers_active_flag )
    NumActiveRefLayerPics = numRefLayerPics
else if( !inter_layer_pred_enabled_flag )                                                (F-53)
    NumActiveRefLayerPics = 0
```

```
else if( max_one_active_ref_layer_flag || NumDirectRefLayers[ nuh_layer_id ] = = 1 )
    NumActiveRefLayerPics = 1
else
    NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1
```

All slices of a coded picture shall have the same value of NumActiveRefLayerPics.

inter_layer_pred_layer_idc[i] specifies the variable, RefPicLayerId[i], representing the nuh_layer_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the inter_layer_pred_layer_idc[i] syntax element is Ceil(Log2(NumDirectRefLayers[nuh_layer_id])) bits. The value of inter_layer_pred_layer_idc[i] shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] − 1, inclusive. When i is greater than 0, inter_layer_pred_layer_idc[i] shall be greater than inter_layer_pred_layer_idc[i − 1]. When not present, the value of inter_layer_pred_layer_idc[i] is inferred to be equal to refLayerPicIdc[i].

The variables RefPicLayerId[i] for all values of i in the range of 0 to NumActiveRefLayerPics − 1, inclusive, are derived as follows:

```
for( i = 0; i < NumActiveRefLayerPics; i++ )                                        (F-54)
    RefPicLayerId[ i ] = IdDirectRefLayer[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]
```

It is a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics − 1, inclusive, either of the following two conditions shall be true:

– The value of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]][LayerIdxInVps[nuh_layer_id]] is greater than TemporalId.

– The values of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]][LayerIdxInVps[nuh_layer_id]] and TemporalId are both equal to 0 and the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i] is an IRAP picture.

poc_reset_idc equal to 0 specifies that neither the most significant bits nor the least significant bits of the picture order count value for the current picture are reset. poc_reset_idc equal to 1 specifies that only the most significant bits of the picture order count value for the current picture may be reset. poc_reset_idc equal to 2 specifies that both the most significant bits and the least significant bits of the picture order count value for the current picture may be reset. poc_reset_idc equal to 3 specifies that either only the most significant bits or both the most significant bits and the least significant bits of the picture order count value for the current picture may be reset and additional picture order count information is signalled. When not present, the value of poc_reset_idc is inferred to be equal to 0.

It is a requirement of bitstream conformance that the following constraints apply:

– The value of poc_reset_idc shall not be equal to 1 or 2 for a RASL, RADL, or SLNR picture or a picture that has TemporalId greater than 0, or a picture that has discardable_flag equal to 1.

– The value of poc_reset_idc of all coded pictures that are present in the bitstream in an access unit shall be the same.

– When the picture in an access unit with nuh_layer_id equal to 0 is an IRAP picture and vps_base_layer_internal_flag is equal to 1 and there is at least one other picture in the same access unit that is not an IRAP picture, the value of poc_reset_idc shall be equal to 1 or 2 for all pictures in the access unit.

– When there is at least one picture that has nuh_layer_id greater than 0 and that is an IDR picture with a particular value of nal_unit_type in an access unit and there is at least one other coded picture that is present in the bitstream in the same access unit with a different value of nal_unit_type, the value of poc_reset_idc shall be equal to 1 or 2 for all pictures in the access unit.

– The value of poc_reset_idc of a CRA or BLA picture shall be less than 3.

– When the picture with nuh_layer_id equal to 0 in an access unit is an IDR picture and vps_base_layer_internal_flag is equal to 1 and there is at least one non-IDR picture in the same access unit, the value of poc_reset_idc shall be equal to 2 for all pictures in the access unit.

– When the picture with nuh_layer_id equal to 0 in an access unit is not an IDR picture and vps_base_layer_internal_flag is equal to 1, the value of poc_reset_idc shall not be equal to 2 for any picture in the access unit.

– When poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]] is equal to 1 and slice_pic_order_cnt_lsb is greater than 0, the value of poc_reset_idc shall not be equal to 2.

The value of poc_reset_idc of an access unit is the value of poc_reset_idc of the pictures in the access unit.

NOTE 3 – Encoders should be cautious in setting the value of poc_reset_idc of pictures when there is a picture of a layer not present in an access unit or there is a picture with discardable_flag equal to 1 present in an access unit, to ensure that the derived picture

order count values of pictures within an access unit are cross-layer aligned. Basically, such cases should be treated similarly as access units with non-cross-layer-aligned IRAP pictures in terms of whether POC resetting is needed.

NOTE 4 – As specified in clause F.8.3.1, the most significant bits of the PicOrderCntVal of an IDR picture with poc_reset_idc equal to 0 are set equal to 0. As specified for the decoder conformance of output order DPB in clause F.13.5.2.2, an IDR picture with nuh_layer_id equal to SmallestLayerId does not cause the output of earlier access units, in decoding order, unless the IDR picture is a POC resetting picture, activates a new VPS, or causes NoClrasOutputFlag to be derived to be equal to 1. Encoders should be cautious when using poc_reset_idc equal to 0 with IDR pictures to maintain the desired picture output order and to obey the constraints on the consistent relation of output times to picture order counts as well as the constraints on the output order of two pictures in different CVSs.

poc_reset_period_id identifies a POC resetting period. When not present, the value of poc_reset_period_id is inferred as follows:

– If there is no picture that is in the same layer as the current picture, precedes the current picture in decoding order, and has poc_reset_period_id present in the slice segment header, the value of poc_reset_period_id is inferred to be equal to 0.

– Otherwise, the value of poc_reset_period_id is inferred to be equal to poc_reset_period_id of the last of such pictures, in decoding order, that are in the same layer as the current picture, precede the current picture in decoding order, and has poc_reset_period_id present in the slice segment header.

NOTE 5 – It is not prohibited for multiple pictures in a layer to have the same value of poc_reset_period_id and to have poc_reset_idc equal to 1 or 2 unless such pictures occur in two consecutive access units in decoding order. To minimize the likelihood of such two pictures appearing in the bitstream due to picture losses, bitstream extraction, seeking, or splicing operations, encoders should set the value of poc_reset_period_id to be a random value for each POC resetting period (subject to the constraints specified above).

It is a requirement of bitstream conformance that the following constraints apply:

– There shall be no two pictures consecutive in decoding order in the same layer that have the same value of poc_reset_period_id and poc_reset_idc equal to 1 or 2.

– One POC resetting period shall not include more than one access unit with poc_reset_idc equal to 1 or 2.

– An access unit with poc_reset_idc equal to 1 or 2 shall be the first access unit in a POC resetting period.

– A picture that follows, in decoding order, the first POC resetting picture among all layers of a POC resetting period in decoding order shall not precede, in output order, another picture in any layer that precedes the first POC resetting picture in decoding order.

– The value of poc_reset_period_id shall be the same for all pictures in an access unit.

full_poc_reset_flag equal to 1 specifies that both the most significant bits and the least significant bits of the picture order count value for the current picture are reset when the previous picture in decoding order in the same layer does not belong to the same POC resetting period. full_poc_reset_flag equal to 0 specifies that only the most significant bits of the picture order count value for the current picture are reset when the previous picture in decoding order in the same layer does not belong to the same POC resetting period.

It is a requirement of bitstream conformance that when the value of poc_reset_idc of pictures in the first access unit in the same POC resetting period is equal to 1 or 2, the value of full_poc_reset_flag, when present, shall be equal to poc_reset_idc − 1.

poc_lsb_val specifies a value that may be used to derive the picture order count of the current picture. The length of the poc_lsb_val syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits.

When poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]] is equal to 1 and full_poc_reset_flag is equal to 1, the value of poc_lsb_val shall be equal to 0.

It is a requirement of bitstream conformance that, when poc_reset_idc is equal to 3, and the previous picture picA in decoding order that is in the same layer as the current picture, that has poc_reset_idc equal to 1 or 2, and that belongs to the same POC resetting period is present in the bitstream, picA shall be the same picture as the previous picture in decoding order that is in the same layer as the current picture, that is not a RASL, RADL, or SLNR picture, and that has TemporalId equal to 0 and discardable_flag equal to 0, and the value of poc_lsb_val of the current picture shall be equal to the value of slice_pic_order_cnt_lsb of picA.

The variable PocMsbValRequiredFlag is derived as follows:

$$\begin{aligned}\text{PocMsbValRequiredFlag} = \text{CraOrBlaPicFlag}\ \&\&\ (\ !\text{vps\_poc\_lsb\_aligned\_flag}\ \|\ \\ (\ \text{vps\_poc\_lsb\_aligned\_flag}\ \&\&\ \text{NumDirectRefLayers}[\ \text{nuh\_layer\_id}\ ]\ ==\ 0\ )\ )\end{aligned} \quad \text{(F-55)}$$

poc_msb_cycle_val_present_flag equal to 1 specifies that poc_msb_cycle_val is present. poc_msb_cycle_val_present_flag equal to 0 specifies that poc_msb_cycle_val is not present. When not present, the value of poc_msb_cycle_val_present_flag is inferred as follows:

– If slice_segment_header_extension_length is equal to 0, the value of poc_msb_cycle_val_present_flag is inferred to be equal to 0.

– Otherwise, if PocMsbValRequiredFlag is equal to 1, the value of poc_msb_cycle_val_present_flag is inferred to be equal to 1.

– Otherwise, the value of poc_msb_cycle_val_present_flag is inferred to be equal to 0.

NOTE 6 – When the current picture is an IDR picture with nuh_layer_id equal to 0 and is a POC resetting picture, vps_poc_lsb_aligned_flag is equal to 1 and NumPredictedLayers[0] is greater than 0, poc_msb_cycle_val_present_flag should be equal to 1.

poc_msb_cycle_val specifies the value that may be used to derive the picture order count value of the current picture or used to derive the value used to decrement the picture order count values of previously decoded pictures in the same layer as the current picture. When vps_poc_lsb_aligned_flag is equal to 1, poc_msb_cycle_val may also specify the value that is used to derive the value used to decrement the picture order count values of previously decoded pictures of the predicted layers of the current layer. The value of poc_msb_cycle_val shall be in the range of 0 to $2^{32 - \text{log2\_max\_pic\_order\_cnt\_lsb\_minus4} - 4}$, inclusive.

NOTE 7 – For a picture picA in layer layerA, let msbAnchorPic of the picture picA be the previous POC resetting picture in the layer layerA or the previous IDR picture that belongs to the layer layerA and has poc_msb_cycle_val_present_flag equal to 0, whichever is closer, in decoding order, to the picture picA. The value of poc_msb_cycle_val is set as follows:

- If vps_poc_lsb_aligned_flag is equal to 0, the following applies:
  - If either of the following conditions is true, the value of poc_msb_cycle_val can be any value in the allowed range:
    - msbAnchorPic of the current picture is not present.
    - the current picture is the first picture in the current layer that belongs to or follows an access unit that contains an IRAP picture with nuh_layer_id equal to SmallestLayerId and NoClrasOutputFlag equal to 1.
  - Otherwise, if the current picture is a POC resetting picture, the value of poc_msb_cycle_val is equal to the difference between the values of the most significant bits of the picture order counts of the current picture, as if there was no POC reset operation for the current picture, and msbAnchorPic of the current picture.
  - Otherwise (the current picture is not a POC resetting picture), the value of poc_msb_cycle_val is equal to the difference between the values of the most significant bits of the picture order count values of the current picture and msbAnchorPic of the current picture.
- Otherwise (vps_poc_lsb_aligned_flag is equal to 1), the following applies:
  - If the current picture is an IDR picture with nuh_layer_id equal to 0 and is a POC resetting picture, encoders should set the value of poc_msb_cycle_val as follows:
    - If picture picB other than the current picture is present in the current access unit, the difference between the most significant bits of the picture order counts of picB and msbAnchorPic of picB.
    - Otherwise (no picture picB other than the current picture is present in the current access unit), the difference between the most significant bits of the picture order counts of picB as if it would be present in the current access unit and msbAnchorPic of such a picB.
  - Otherwise, if the current picture is the first picture in the POC resetting period and has nuh_layer_id equal to SmallestLayerid, the value of poc_msb_cycle_val is equal to the difference between the values of the most significant bits of the picture order counts of the current picture, as if there was no POC reset operation for the current picture, and msbAnchorPic of the current picture.
  - Otherwise when the current picture is not a POC resetting picture, the value of poc_msb_cycle_val is equal to the difference between the values of the most significant bits of the picture order count values of the current picture and msbAnchorPic of the current picture.

slice_segment_header_extension_data_bit may have any value. Decoders shall ignore the value of slice_segment_header_extension_data_bit. Its value does not affect the decoding process specified in this version of this Specification.

F.7.4.7.2 Reference picture list modification semantics

The specifications in clause 7.4.7.2 apply with following modifications:

– Equation 7-57 specifying the derivation of NumPicTotalCurr is replaced by:

```
NumPicTotalCurr = 0
if( nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP ) {
    for( i = 0; i < NumNegativePics[ CurrRpsIdx ]; i++ )
        if( UsedByCurrPicS0[ CurrRpsIdx ][ i ] )
            NumPicTotalCurr++
    for( i = 0; i < NumPositivePics[ CurrRpsIdx ]; i++ )                (F-56)
        if( UsedByCurrPicS1[ CurrRpsIdx ][ i ] )
            NumPicTotalCurr++
    for( i = 0; i < num_long_term_sps + num_long_term_pics; i++ )
        if( UsedByCurrPicLt[ i ] )
            NumPicTotalCurr++
}
if( pps_curr_pic_ref_enabled_flag )
    NumPicTotalCurr++
NumPicTotalCurr += NumActiveRefLayerPics
```

F.7.4.7.3 Weighted prediction parameters semantics

The specifications in clause 7.4.7.3 apply.

F.7.4.8 Short-term reference picture set semantics

The specifications in clause 7.4.8 apply, with the following additions:

When vps_extension_flag is equal to 1, the value of num_negative_pics shall be in the range of 0 to MaxDpbSize − 1, inclusive, where MaxDpbSize is as specified in clause G.11.2 or H.11.2.

When vps_extension_flag is equal to 1, the value of num_positive_pics shall be in the range of 0 to MaxDpbSize − 1 − num_negative_pics, inclusive, where MaxDpbSize is as specified in clause G.11.2 or H.11.2.

F.7.4.9 Slice segment data semantics

F.7.4.9.1 General slice segment data semantics

The specifications in clause 7.4.9.1 apply.

F.7.4.9.2 Coding tree unit semantics

The specifications in clause 7.4.9.2 apply.

F.7.4.9.3 Sample adaptive offset semantics

The specifications in clause 7.4.9.3 apply.

F.7.4.9.4 Coding quadtree semantics

The specifications in clause 7.4.9.4 apply.

F.7.4.9.5 Coding unit semantics

The specifications in clause 7.4.9.5 apply.

F.7.4.9.6 Prediction unit semantics

The specifications in clause 7.4.9.6 apply.

F.7.4.9.7 PCM sample semantics

The specifications in clause 7.4.9.7 apply.

F.7.4.9.8 Transform tree semantics

The specifications in clause 7.4.9.8 apply.

F.7.4.9.9 Motion vector difference semantics

The specifications in clause 7.4.9.9 apply.

F.7.4.9.10 Transform unit semantics

The specifications in clause 7.4.9.10 apply.

F.7.4.9.11 Residual coding semantics

The specifications in clause 7.4.9.11 apply.

F.7.4.9.12 Cross-component prediction syntax

The specifications in clause 7.4.9.12 apply.

F.7.4.9.13 Palette mode semantics

The specifications in clause 7.4.9.13 apply.

F.7.4.9.14 Delta QP semantics

The specifications in clause 7.4.9.14 apply.

F.7.4.9.15 Chroma QP offset semantics

The specifications in clause 7.4.9.15 apply.

F.8 Decoding process

F.8.1 General decoding process

F.8.1.1 General

Input to this process is a bitstream. Output of this process is a list of decoded pictures.

The decoding process is specified such that all decoders capable of decoding an OLS where each of the output layers and the reference layers of the output layers is indicated to conform to a specified profile, tier and level will produce numerically identical cropped decoded output pictures when invoking the decoding process associated with those profiles for the OLS. Any decoding process that produces identical cropped decoded output pictures to those produced by the process described herein (with the correct output order or output timing, as specified) conforms to the decoding process requirements of this Specification.

The following applies at the beginning of decoding a CVSG, after activating the VPS RBSP that is active for the entire CVSG and before decoding any VCL NAL units of the CVSG:

– If vps_extension( ) is not present in the active VPS or a decoding process specified in this annex is not in use, clause 8.1.2 is invoked with the CVSG as input.

– Otherwise (vps_extension( ) is present in the active VPS and a decoding process specified in this annex is in use), clause F.8.1.2 is invoked with the CVSG as input.

F.8.1.2 CVSG decoding process

Input to this process is a CVSG. Output of this process is a list of decoded pictures.

The variable TargetOlsIdx, which specifies the index to the list of the OLSs specified by the VPS, of the target OLS, is specified as follows:

– If some external means, not specified in this Specification, is available to set TargetOlsIdx, TargetOlsIdx is set by the external means.

– Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause F.13.1, TargetOlsIdx is set as specified in clause F.13.1.

– Otherwise, TargetOlsIdx is set equal to 0.

The variable TargetDecLayerSetIdx, the layer identifier list TargetOptLayerIdList, which specifies the list of nuh_layer_id values, in increasing order of nuh_layer_id values, of the pictures to be output and the layer identifier list TargetDecLayerIdList, which specifies the list of nuh_layer_id values, in increasing order of nuh_layer_id values, of the NAL units to be decoded, are specified as follows:

```
TargetDecLayerSetIdx = OlsIdxToLsIdx[ TargetOlsIdx ]
lsIdx = TargetDecLayerSetIdx
for( i = 0, j = 0, k = 0; i < NumLayersInIdList[ lsIdx ]; i++ ) {
```

```
        if( NecessaryLayerFlag[ TargetOlsIdx ][ i ] )
            TargetDecLayerIdList[ j++ ] = LayerSetLayerIdList[ lsIdx ][ i ]         (F-57)
        if( OutputLayerFlag[ TargetOlsIdx ][ i ] )
            TargetOptLayerIdList[ k++ ] = LayerSetLayerIdList[ lsIdx ][ i ]
    }
```

When TargetOlsIdx is set by external means, it is a requirement for the external means that TargetOlsIdx is constrained as follows:

– When vps_base_layer_available_flag is equal to 0, OlsIdxToLsIdx[TargetOlsIdx] shall be in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive.

– When vps_base_layer_internal_flag is equal to 0, TargetOlsIdx shall be greater than 0.

The variable HighestTid, which identifies the highest temporal sub-layer to be decoded, is specified as follows:

– If some external means, not specified in this Specification, is available to set HighestTid, HighestTid is set by the external means.

– Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause F.13.1 HighestTid is set as specified in clause F.13.1.

– Otherwise, HighestTid is set equal to vps_max_sub_layers_minus1.

The variable SubPicHrdPreferredFlag is either specified by external means, or when not specified by external means, set equal to 0.

The variable SubPicHrdFlag is specified as follows:

– If the decoding process is invoked in a bitstream conformance test as specified in clause F.13.1, SubPicHrdFlag is set as specified in clause F.13.1.

– Otherwise, SubPicHrdFlag is set equal to (SubPicHrdPreferredFlag && sub_pic_hrd_params_present_flag), where sub_pic_hrd_params_present_flag is found in any hrd_parameters( ) syntax structure that applies to at least one bitstream partition of the output layer set identified by TargetOlsIdx.

A bitstream to be decoded, BitstreamToDecode, is specified as follows:

– If TargetDecLayerSetIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 1, the following applies:

  – The sub-bitstream extraction process as specified in clause 10 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.

  – The variable SmallestLayerId is set equal to 0.

– Otherwise, if TargetDecLayerSetIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 0, the following applies:

  – The sub-bitstream extraction process as specified in clause F.10.1 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.

  – The variable SmallestLayerId is set equal to 0.

– Otherwise, if TargetDecLayerSetIdx is greater than vps_num_layer_sets_minus1 and NumLayersInIdList[TargetDecLayerSetIdx] is equal to 1, the following applies:

  – The independent non-base layer rewriting process of clause F.10.2 is applied with the CVSG, HighestTid and TargetDecLayerIdList[0] as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.

  – The variable SmallestLayerId is set equal to 0.

– Otherwise, the following applies:

  – The sub-bitstream extraction process as specified in clause F.10.3 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.

  – The variable SmallestLayerId is set equal to TargetDecLayerIdList[0].

When vps_base_layer_internal_flag is equal to 0, vps_base_layer_available_flag is equal to 1 and TargetDecLayerSetIdx is in the range of 0 to vps_num_layer_sets_minus1, inclusive, the following applies:

– The size of the sub-DPB for the layer with nuh_layer_id equal to 0 is set equal to 1.

– The values of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8, conf_win_left_offset, conf_win_right_offset, conf_win_top_offset and conf_win_bottom_offset for decoded pictures with nuh_layer_id equal to 0 are set equal to the values of pic_width_vps_in_luma_samples, pic_height_vps_in_luma_samples, chroma_format_vps_idc, separate_colour_plane_vps_flag, bit_depth_vps_luma_minus8, bit_depth_vps_chroma_minus8, conf_win_vps_left_offset, conf_win_vps_right_offset, conf_win_vps_top_offset and conf_win_vps_bottom_offset respectively, of the vps_rep_format_idx[0]-th rep_format( ) syntax structure in the active VPS.

– The variable BaseLayerOutputFlag is set equal to (TargetOptLayerIdList[0] = = 0).

  NOTE – The BaseLayerOutputFlag is to be sent by an external means to the base layer decoder for controlling the output of base layer decoded pictures. BaseLayerOutputFlag equal to 1 indicates that the base layer is an output layer. BaseLayerOutputFlag equal to 0 indicates that the base layer is not an output layer.

– The variable LayerInitializedFlag[i] is set equal to 0 for all values of i from 0 to vps_max_layer_id, inclusive, and the variable FirstPicInLayerDecodedFlag[i] is set equal to 0 for all values of i from 0 to vps_max_layer_id, inclusive.

If TargetOlsIdx is equal to 0, clause 8.1.3 is repeatedly invoked for each coded picture in BitstreamToDecode in decoding order and the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause 7.

Otherwise (TargetOlsIdx is greater than 0), clause F.8.1.3 is repeatedly invoked for each coded picture in BitstreamToDecode in decoding order.

F.8.1.3 Common decoding process for a coded picture

The decoding processes specified in the remainder of this clause apply to each coded picture, referred to as the current picture and denoted by the variable CurrPic, in BitstreamToDecode.

Depending on the value of chroma_format_idc, the number of sample arrays of the current picture is as follows:

– If chroma_format_idc is equal to 0, the current picture consists of 1 sample array S_L.

– Otherwise (chroma_format_idc is not equal to 0), the current picture consists of 3 sample arrays S_L, S_{Cb}, S_{Cr}.

When interpreting the semantics of each syntax element in each NAL unit, the term "the bitstream" (or part thereof, e.g., a CVS of the bitstream) refers to BitstreamToDecode (or part thereof).

When vps_base_layer_internal_flag is equal to 0, vps_base_layer_available_flag is equal to 1, TargetDecLayerSetIdx is in the range of 0 to vps_num_layer_sets_minus1, inclusive, TemporalId is less than or equal to sub_layers_vps_max_minus1[0], and the current picture is the first coded picture of an access unit, clause F.8.1.8 is invoked prior to decoding the current picture.

When the current picture is an IRAP picture, the variable HandleCraAsBlaFlag is derived as specified in the following:

– If some external means not specified in this Specification is available to set the variable HandleCraAsBlaFlag to a value for the current picture, the variable HandleCraAsBlaFlag is set equal to the value provided by the external means.

– Otherwise, the variable HandleCraAsBlaFlag is set equal to 0.

When the current picture is an IRAP picture and has nuh_layer_id equal to SmallestLayerId, the following applies:

– The variable NoClrasOutputFlag is specified as follows:

  – If the current picture is the first picture in the bitstream, NoClrasOutputFlag is set equal to 1.

  – Otherwise, if the current picture is included in the first access unit that follows an access unit including an end of sequence NAL unit with nuh_layer_id equal to SmallestLayerId or 0 in decoding order, NoClrasOutputFlag is set equal to 1.

  – Otherwise, if the current picture is a BLA picture or a CRA picture with HandleCraAsBlaFlag equal to 1, NoClrasOutputFlag is set equal to 1.

  – Otherwise, if the current picture is an IDR picture with cross_layer_bla_flag is equal to 1, NoClrasOutputFlag is set equal to 1.

  – Otherwise, if some external means, not specified in this Specification, is available to set NoClrasOutputFlag, NoClrasOutputFlag is set by the external means.

  – Otherwise, NoClrasOutputFlag is set equal to 0.

– When NoClrasOutputFlag is equal to 1, the variable LayerInitializedFlag[i] is set equal to 0 for all values of i from 0 to vps_max_layer_id, inclusive, and the variable FirstPicInLayerDecodedFlag[i] is set equal to 0 for all values of i from 0 to vps_max_layer_id, inclusive.

The variables LayerResetFlag and dolLayerId are derived as follows:

– If the current picture is an IRAP picture and has nuh_layer_id nuhLayerId greater than SmallestLayerId, the following applies:

  – If the current picture is the first picture, in decoding order, that follows an end of sequence NAL unit with nuh_layer_id equal to nuhLayerId, LayerResetFlag is set equal to 1 and dolLayerId is set equal to the nuh_layer_id value of the current NAL unit.

  – Otherwise, if the current picture is a CRA picture with HandleCraAsBlaFlag equal to 1, an IDR picture with cross_layer_bla_flag is equal to 1 or a BLA picture, LayerResetFlag is set equal to 1 and dolLayerId is set equal to the nuh_layer_id value of the current NAL unit.

  – Otherwise, LayerResetFlag is set equal to 0.

  NOTE 1 – An end of sequence NAL unit, a CRA picture with HandleCraAsBlaFlag equal to 1, an IDR picture with cross_layer_bla_flag equal to 1, or a BLA picture, each with nuh_layer_id nuhLayerId greater than SmallestLayerId, may be present to indicate a discontinuity of the layer with nuh_layer_id equal to nuhLayerId and its predicted layers.

  – When LayerResetFlag is equal to 1, the following applies:

    – The values of LayerInitializedFlag and FirstPicInLayerDecodedFlag are updated as follows:

```
for( i = 0; i < NumPredictedLayers[ dolLayerId ]; i++ ) {
    iLayerId = IdPredictedLayer[ dolLayerId ][ i ]                    (F-58)
    LayerInitializedFlag[ iLayerId ] = 0
    FirstPicInLayerDecodedFlag[ iLayerId ] = 0
}
```

    – Each picture that is in the DPB and has nuh_layer_id equal to dolLayerId is marked as "unused for reference".

    – When NumPredictedLayers[dolLayerId] is greater than 0, each picture that is in the DPB and has nuh_layer_id equal to any value of IdPredictedLayer[dolLayerId][i] for the values of i in the range of 0 to NumPredictedLayers[dolLayerId] − 1, inclusive, is marked as "unused for reference".

– Otherwise, LayerResetFlag is set equal to 0.

When the current picture is an IRAP picture, the following applies:

– If the current picture with a particular value of nuh_layer_id is an IDR picture, a BLA picture, the first picture with that particular value of nuh_layer_id in the bitstream in decoding order or the first picture with that particular value of nuh_layer_id that follows an end of sequence NAL unit with that particular value of nuh_layer_id in decoding order, the variable NoRaslOutputFlag is set equal to 1.

– Otherwise, if LayerInitializedFlag[nuh_layer_id] is equal to 0 and LayerInitializedFlag[refLayerId] is equal to 1 for all values of refLayerId equal to IdDirectRefLayer[nuh_layer_id][j], where j is in the range of 0 to NumDirectRefLayers[nuh_layer_id] − 1, inclusive, the variable NoRaslOutputFlag is set equal to 1.

– Otherwise, the variable NoRaslOutputFlag is set equal to HandleCraAsBlaFlag.

When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and one of the following conditions is true, LayerInitializedFlag[nuh_layer_id] is set equal to 1:

– nuh_layer_id is equal to 0.

– LayerInitializedFlag[nuh_layer_id] is equal to 0 and NumDirectRefLayers[nuh_layer_id] is equal to 0.

– LayerInitializedFlag[nuh_layer_id] is equal to 0 and LayerInitializedFlag[refLayerId] is equal to 1 for all values of refLayerId equal to IdDirectRefLayer[nuh_layer_id][j], where j is in the range of 0 to NumDirectRefLayers[nuh_layer_id] − 1, inclusive.

Depending on the value of separate_colour_plane_flag, the decoding process is structured as follows:

– If separate_colour_plane_flag is equal to 0, the following decoding process is invoked a single time with the current picture being the output.

– Otherwise (separate_colour_plane_flag is equal to 1), the following decoding process is invoked three times. Inputs to the decoding process are all NAL units of the coded picture with identical value of colour_plane_id. The decoding

process of NAL units with a particular value of colour_plane_id is specified as if only a CVS with monochrome colour format with that particular value of colour_plane_id would be present in the bitstream. The output of each of the three decoding processes is assigned to one of the 3 sample arrays of the current picture, with the NAL units with colour_plane_id equal to 0, 1 and 2 being assigned to S_L, S_{Cb} and S_{Cr}, respectively.

NOTE 2 – The variable ChromaArrayType is derived as equal to 0 when separate_colour_plane_flag is equal to 1 and chroma_format_idc is equal to 3. In the decoding process, the value of this variable is evaluated resulting in operations identical to that of monochrome pictures (when chroma_format_idc is equal to 0).

The following applies for the decoding of the current picture:

– If the current picture has nuh_layer_id equal to 0, the following applies:

  – The decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause F.7.

  – The variables NumActiveRefLayerPics, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0.

  – The decoding process for a coded picture with nuh_layer_id equal to 0 as specified in clause F.8.1.4 is invoked.

– Otherwise, the following applies:

  – When vps_base_layer_internal_flag is equal to 0, vps_base_layer_available_flag is equal to 1, TargetDecLayerSetIdx is in the range of 0 to vps_num_layer_sets_minus1, inclusive, TemporalId is less than or equal to sub_layers_vps_max_minus1[0], the current picture is the first coded picture of an access unit and a decoded picture with nuh_layer_id equal to 0 is provided by external means for the current access unit, clause F.8.1.9 is invoked after the decoding of the slice segment header of the first slice segment, in decoding order, of the current picture, but prior to decoding any slice segment of the first coded picture of the access unit.

  – For the decoding of the slice segment header of the first slice segment, in decoding order, of the current picture, the decoding process for starting the decoding of a coded picture with nuh_layer_id greater than 0 specified in clause F.8.1.5 is invoked.

  – Let lIdx be equal to such value for which the nuh_layer_id value of the current picture is equal to LayerSetLayerIdList[OlsIdxToLsIdx[TargetOlsIdx]][lIdx].

  – If general_profile_idc in the profile_tier_level( ) syntax structure VpsProfileTierLevel[profile_tier_level_idx[TargetOlsIdx][lIdx]] is equal to 6, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause G.7 and the decoding process of clause G.8.1.2 is invoked.

  – Otherwise, if general_profile_idc in the profile_tier_level( ) syntax structure VpsProfileTierLevel[profile_tier_level_idx[TargetOlsIdx][lIdx]] is equal to 7 or 10, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause H.7 and the decoding process of clause H.8.1.2 is invoked.

  – Otherwise, if general_profile_idc in the profile_tier_level( ) syntax structure VpsProfileTierLevel[profile_tier_level_idx[TargetOlsIdx][lIdx]] is equal to 8, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause I.7 and the decoding process of clause I.8.1.2 is invoked.

  – Otherwise (the current picture belongs to an independent non-base layer), the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause F.7 and the decoding process of clause 8.1.3 is invoked by changing the references to clauses 8.2, 8.3, 8.3.1, 8.3.2, 8.3.3, 8.3.4, 8.4, 8.5, 8.6, and 8.7 with clauses F.8.2, F.8.3, F.8.3.1, F.8.3.2, F.8.3.3, F.8.3.4, F.8.4, F.8.5, F.8.6, and F.8.7, respectively.

  – After all slices of the current picture have been decoded, the decoding process for ending the decoding of a coded picture with nuh_layer_id greater than 0 specified in clause F.8.1.6 is invoked.

When the current picture is the last coded picture in an access unit in BitstreamToDecode, after the decoding of the current picture, prior to the decoding of the next picture, the following applies:

– PicOutputFlag is updated as follows:

  – If alt_output_layer_flag[TargetOlsIdx] is equal to 1 and the current access unit either does not contain a picture at the output layer or contains a picture at the output layer that has PicOutputFlag equal to 0, the following applies:

    – The list nonOutputLayerPictures is set to be the list of the pictures of the access unit with PicOutputFlag equal to 1 and with nuh_layer_id values among the nuh_layer_id values of the reference layers of the output layer.

- When the list nonOutputLayerPictures is not empty, the picture with the highest nuh_layer_id value among the list nonOutputLayerPictures is removed from the list nonOutputLayerPictures.
- PicOutputFlag for each picture that is included in the list nonOutputLayerPictures is set equal to 0.

– Otherwise, PicOutputFlag for pictures that are not included in an output layer is set equal to 0.

- When vps_base_layer_internal_flag is equal to 0, vps_base_layer_available_flag is equal to 1 and TargetDecLayerSetIdx is in the range of 0 to vps_num_layer_sets_minus1, inclusive, the following applies:
  - If BaseLayerOutputFlag is equal to 0, the following applies:
    - If alt_output_layer_flag[TargetOlsIdx] is equal to 1, the base layer is a reference layer of the output layer, the access unit does not contain a picture at the output layer or contains a picture at the output layer that has PicOutputFlag equal to 0, and the access unit does not contain a picture at any other reference layer of the output layer, BaseLayerPicOutputFlag is set equal to 1.
    - Otherwise, BaseLayerPicOutputFlag is set equal to 0.
  - Otherwise, BaseLayerPicOutputFlag is set equal to 1.

  NOTE 3 – The BaseLayerPicOutputFlag for each access unit is to be sent by an external means to the base layer decoder for controlling the output of base layer decoded pictures. BaseLayerPicOutputFlag equal to 1 for an access unit specifies that the base layer picture of the access unit is to be output. BaseLayerPicOutputFlag equal to 0 for an access unit specifies that the base layer picture of the access unit is not to be output.
  - The sub-DPB for the layer with nuh_layer_id equal to 0 is set to be empty.
- The variable AuOutputFlag that is associated with the current access unit is derived as follows:
  - If at least one picture in the current access unit has PicOutputFlag equal to 1, AuOutputFlag is set equal to 1.
  - Otherwise, AuOutputFlag is set equal to 0.
- The variable PicLatencyCount that is associated with the current access unit is set equal to 0.
- When AuOutputFlag of the current access unit is equal to 1, for each access unit in the DPB that has at least one picture marked as "needed for output" and follows the current access unit in output order, the associated variable PicLatencyCount is set equal to PicLatencyCount + 1.

F.8.1.4 Decoding process for a coded picture with nuh_layer_id equal to 0

The specifications in clause 8.1.3 apply with the following changes:

- Replace the references to clauses 8.2, 8.3, 8.3.1, 8.3.2, 8.3.3, 8.3.4, 8.4, 8.5, 8.6 and 8.7 with clauses F.8.2, F.8.3, F.8.3.1, F.8.3.2, F.8.3.3, F.8.3.4, F.8.4, F.8.5, F.8.6 and F.8.7, respectively.
- At the end of the clause, add item 5 as follows:

  5. When FirstPicInLayerDecodedFlag[0] is equal to 0, FirstPicInLayerDecodedFlag[0] is set equal to 1.

F.8.1.5 Decoding process for starting the decoding of a coded picture with nuh_layer_id greater than 0

Each picture referred to in this clause is a complete coded picture.

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in clause F.8.2.
2. The processes in clause F.8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:
   - Variables and functions relating to picture order count are derived in clause F.8.3.1. This needs to be invoked only for the first slice segment of a picture. It is a requirement of bitstream conformance that PicOrderCntVal of each picture in an access unit shall have the same value during and at the end of decoding of the access unit.

     NOTE 1 – When the current picture is or succeeds, in decoding order, an IDR picture with nuh_layer_id equal to 0, PicOrderCntVal of a base layer picture picA preceding, in decoding order, the IDR picture with nuh_layer_id equal to 0 may or may not be equal to PicOrderCntVal of the pictures with nuh_layer_id greater than 0 in the access unit containing picA.
   - The decoding process for RPS in clause F.8.3.2 is invoked, wherein only reference pictures with nuh_layer_id equal to that of CurrPic may be marked as "unused for reference" or "used for long-term

reference" and any picture with a different value of nuh_layer_id is not marked. This needs to be invoked only for the first slice segment of a picture.

– A picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture after the invocation of the in-loop filter process as specified in clause F.8.7. This version of the current decoded picture is referred to as the current decoded picture after the invocation of the in-loop filter process. When TwoVersionsOfCurrDecPicFlag is equal to 0 and pps_curr_pic_ref_enabled_flag is equal to 1, this picture storage buffer is marked as "used for long-term reference". When TwoVersionsOfCurrDecPicFlag is equal to 1, another picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture immediately before the invocation of the in-loop filter process as specified in clause F.8.7, and is marked as "used for long-term reference". This version of the current decoded picture is referred to as the current decoded picture before the invocation of the in-loop filter process. This needs to be invoked only for the first slice segment of a picture.

NOTE 2 – When TwoVersionsOfCurrDecPicFlag is equal to 0, there is only one version of the current decoded picture. In this case, if pps_curr_pic_ref_enabled_flag is equal to 1, the current decoded picture is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture, otherwise it is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture. When TwoVersionsOfCurrDecPicFlag is equal to 1, there are two versions of the current decoded picture, one of which is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "unused for reference" at the end of the decoding of the current picture, and the other version is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture.

– When FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 0, the decoding process for generating unavailable reference pictures for pictures first in decoding order within a layer specified in clause F.8.1.7 is invoked, which needs to be invoked only for the first slice segment of a picture.

– When FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 1 and the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, the decoding process for generating unavailable reference pictures specified in clause F.8.3.3 is invoked, which needs to be invoked only for the first slice segment of a picture.

F.8.1.6 Decoding process for ending the decoding of a coded picture with nuh_layer_id greater than 0

The marking of decoded pictures is modified as specified in the following:

```
for( i = 0; i < NumActiveRefLayerPics0; i++ )
    RefPicSetInterLayer0[ i ] is marked as "used for short-term reference"      (F-59)
for( i = 0; i < NumActiveRefLayerPics1; i++ )
    RefPicSetInterLayer1[ i ] is marked as "used for short-term reference"
```

PicOutputFlag is set as follows:

– If LayerInitializedFlag[nuh_layer_id] is equal to 0, PicOutputFlag is set equal to 0.

– Otherwise, if the current picture is a RASL picture and NoRaslOutputFlag of the associated IRAP picture is equal to 1, PicOutputFlag is set equal to 0.

– Otherwise, PicOutputFlag is set equal to pic_output_flag.

The current decoded picture after the invocation of the in-loop filter process as specified in clause 8.7 is marked as "used for short-term reference".

When FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 0, FirstPicInLayerDecodedFlag[nuh_layer_id] is set equal to 1.

F.8.1.7 Decoding process for generating unavailable reference pictures for pictures first in decoding order within a layer

This process is invoked for a picture with nuh_layer_id equal to layerId, when FirstPicInLayerDecodedFlag[layerId] is equal to 0.

NOTE – The entire specification of the decoding process for CL-RAS pictures is included only for purposes of specifying constraints on the allowed syntax content of such CL-RAS pictures. During the decoding process, any CL-RAS pictures may be ignored, as these pictures are not specified for output and have no effect on the decoding process of any other pictures that are specified for output. However, in the HRD operations as specified in clause F.13, CL-RAS pictures may need to be taken into consideration in the derivation of CPB arrival and removal times.

When this process is invoked, the following applies:

- For each RefPicSetStCurrBefore[i], with i in the range of 0 to NumPocStCurrBefore − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
  - The value of PicOrderCntVal for the generated picture is set equal to PocStCurrBefore[i].
  - The value of PicOutputFlag for the generated picture is set equal to 0.
  - The generated picture is marked as "used for short-term reference".
  - RefPicSetStCurrBefore[i] is set to be the generated reference picture.
  - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id.
- For each RefPicSetStCurrAfter[i], with i in the range of 0 to NumPocStCurrAfter − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
  - The value of PicOrderCntVal for the generated picture is set equal to PocStCurrAfter[i].
  - The value of PicOutputFlag for the generated picture is set equal to 0.
  - The generated picture is marked as "used for short-term reference".
  - RefPicSetStCurrAfter[i] is set to be the generated reference picture.
  - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id.
- For each RefPicSetStFoll[i], with i in the range of 0 to NumPocStFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
  - The value of PicOrderCntVal for the generated picture is set equal to PocStFoll[i].
  - The value of PicOutputFlag for the generated picture is set equal to 0.
  - The generated picture is marked as "used for short-term reference".
  - RefPicSetStFoll[i] is set to be the generated reference picture.
  - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id.
- For each RefPicSetLtCurr[i], with i in the range of 0 to NumPocLtCurr − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
  - The value of PicOrderCntVal for the generated picture is set equal to PocLtCurr[i].
  - The value of slice_pic_order_cnt_lsb for the generated picture is inferred to be equal to (PocLtCurr[i] & (MaxPicOrderCntLsb − 1)).
  - The value of PicOutputFlag for the generated picture is set equal to 0.
  - The generated picture is marked as "used for long-term reference".
  - RefPicSetLtCurr[i] is set to be the generated reference picture.
  - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id.
- For each RefPicSetLtFoll[i], with i in the range of 0 to NumPocLtFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
  - The value of PicOrderCntVal for the generated picture is set equal to PocLtFoll[i].
  - The value of slice_pic_order_cnt_lsb for the generated picture is inferred to be equal to (PocLtFoll[i] & (MaxPicOrderCntLsb − 1)).
  - The value of PicOutputFlag for the generated picture is set equal to 0.
  - The generated picture is marked as "used for long-term reference".
  - RefPicSetLtFoll[i] is set to be the generated reference picture.
  - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id.

F.8.1.8 Initialization process for an external base layer picture

A decoded picture with nuh_layer_id equal to 0 may be provided by external means. When not provided, no picture with nuh_layer_id equal to 0 is used for inter-layer prediction for the current access unit. When provided, the following applies:

– The variable LayerInitializedFlag[0] is set equal to 1 and the variable FirstPicInLayerDecodedFlag[0] is set equal to 1.

– The following information of the picture with nuh_layer_id equal to 0 for the access unit is provided by external means:

  – The decoded sample values (1 sample array S_L if chroma_format_idc is equal to 0 or 3 sample arrays S_L, S_{Cb}, and S_{Cr} otherwise)

  – The value of the variable BlIrapPicFlag, and when BlIrapPicFlag is equal to 1, the value of nal_unit_type of the decoded picture

    – BlIrapPicFlag equal to 1 specifies that the decoded picture is an IRAP picture. BlIrapPicFlag equal to 0 specifies that the decoded picture is a non-IRAP picture.

    – The provided value of nal_unit_type of the decoded picture shall be equal to IDR_W_RADL, CRA_NUT, or BLA_W_LP.

      – nal_unit_type equal to IDR_W_RADL specifies that the decoded picture is an IDR picture.

      – nal_unit_type equal to CRA_NUT specifies that the decoded picture is a CRA picture.

      – nal_unit_type equal to BLA_W_LP specifies that the decoded picture is a BLA picture.

– When BlIrapPicFlag of the picture with nuh_layer_id equal to 0 is equal to 1, the following applies for the decoded picture with nuh_layer_id equal to 0 for the access unit:

  – The variable NoRaslOutputFlag is specified as follows:

    – If nal_unit_type is IDR_W_RADL or BLA_W_LP, the variable NoRaslOutputFlag is set equal to 1.

    – Otherwise, if the current access unit is the first access unit in the bitstream in decoding order, the variable NoRaslOutputFlag is set equal to 1.

    – Otherwise, the variable NoRaslOutputFlag is set equal to 0.

  – The variable NoClrasOutputFlag is specified as follows:

    – If the current access unit is the first access unit in the bitstream, NoClrasOutputFlag is set equal to 1.

    – Otherwise, if nal_unit_type is equal to BLA_W_LP, NoClrasOutputFlag is set equal to 1.

    – Otherwise, if some external means, not specified in this Specification, is available to set NoClrasOutputFlag, NoClrasOutputFlag is set by the external means.

    – Otherwise, NoClrasOutputFlag is set equal to 0.

  – When NoClrasOutputFlag is equal to 1, the variable LayerInitializedFlag[i] is set equal to 0 for all values of i from 1 to vps_max_layer_id, inclusive, and the variable FirstPicInLayerDecodedFlag[i] is set equal to 0 for all values of i from 1 to vps_max_layer_id, inclusive.

F.8.1.9 Decoding process for an external base layer picture

The following applies for the decoded picture with nuh_layer_id equal to 0 for the access unit:

– TemporalId and PicOrderCntVal of the decoded picture with nuh_layer_id equal to 0 are set equal to the TemporalId and PicOrderCntVal, respectively, of any picture with nuh_layer_id greater than 0 in the access unit.

  NOTE – The constraint on the value of TemporalId being required to be equal to 0 for IRAP pictures also applies to pictures with nuh_layer_id equal to 0 when vps_base_layer_internal_flag is equal to 0.

– The decoded picture with nuh_layer_id equal to 0 is stored in the sub-DPB for the layer with nuh_layer_id equal to 0 and is marked as "used for long-term reference".

F.8.2 NAL unit decoding process

The specifications in clause 8.2 apply.

F.8.3 Slice decoding processes

F.8.3.1 Decoding process for picture order count

Output of this process is PicOrderCntVal, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction and for decoder conformance checking (see clause F.13.5).

Each coded picture is associated with a picture order count variable, denoted as PicOrderCntVal.

When the current picture is the first picture among all layers of a POC resetting period, the variable PocDecrementedInDPBFlag[i] is set equal to 0 for each value of i in the range of 0 to 62, inclusive.

The variable pocResettingFlag is derived as follows:

– If the current picture is a POC resetting picture, the following applies:
  – If vps_poc_lsb_aligned_flag is equal to 0, pocResettingFlag is set equal to 1.
  – Otherwise, if PocDecrementedInDPBFlag[nuh_layer_id] is equal to 1, pocResettingFlag is set equal to 0.
  – Otherwise, pocResettingFlag is set equal to 1.
– Otherwise, pocResettingFlag is set equal to 0.

The list affectedLayerList is derived as follows:

– If vps_poc_lsb_aligned_flag is equal to 0, affectedLayerList consists of the nuh_layer_id of the current picture.
– Otherwise, affectedLayerList consists of the nuh_layer_id of the current picture and the nuh_layer_id values equal to IdPredictedLayer[currNuhLayerId][j] for all values of j in the range of 0 to NumPredictedLayers[currNuhLayerId] − 1, inclusive, where currNuhLayerId is the nuh_layer_id value of the current picture.

Depending on pocResettingFlag, the following applies:

– If pocResettingFlag is equal to 1, the following applies:
  – When FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 1, the following applies:
    – The variables pocMsbDelta, pocLsbDelta and DeltaPocVal are derived as follows:

```
if( poc_reset_idc  = =  3 )
    pocLsbVal = poc_lsb_val
else
    pocLsbVal = slice_pic_order_cnt_lsb
if( poc_msb_cycle_val_present_flag )
    pocMsbDelta = poc_msb_cycle_val * MaxPicOrderCntLsb                    (F-60)
else {
    prevPicOrderCntLsb = PrevPicOrderCnt[ nuh_layer_id ] & ( MaxPicOrderCntLsb − 1 )
    prevPicOrderCntMsb = PrevPicOrderCnt[ nuh_layer_id ] − prevPicOrderCntLsb
    pocMsbDelta = GetCurrMsb( pocLsbVal, prevPicOrderCntLsb, prevPicOrderCntMsb,
            MaxPicOrderCntLsb )
}
if( poc_reset_idc  = =  2  | |  ( poc_reset_idc  = =  3  &&  full_poc_reset_flag ) )
    pocLsbDelta = pocLsbVal
else
    pocLsbDelta = 0
DeltaPocVal = pocMsbDelta + pocLsbDelta
```

    – The PicOrderCntVal of each picture that has nuh_layer_id value nuhLayerId for which PocDecrementedInDPBFlag[nuhLayerId] is equal to 0 and that is equal to any value in affectedLayerList is decremented by DeltaPocVal.
    – PocDecrementedInDPBFlag[nuhLayerId] is set equal to 1 for each value of nuhLayerId included in affectedLayerList.
  – The PicOrderCntVal of the current picture is derived as follows:

```
if( poc_reset_idc  = =  1 )
    PicOrderCntVal = slice_pic_order_cnt_lsb
else if( poc_reset_idc  = =  2 )
    PicOrderCntVal = 0
else {
```

```
        PicOrderCntMsb =                                                        (F-61)
            GetCurrMsb( slice_pic_order_cnt_lsb, full_poc_reset_flag ? 0 : poc_lsb_val,
                    0, MaxPicOrderCntLsb )
        PicOrderCntVal = PicOrderCntMsb + slice_pic_order_cnt_lsb
    }
```

– Otherwise (pocResettingFlag is equal to 0), the following applies:

  – The PicOrderCntVal of the current picture is derived as follows:

```
if( poc_msb_cycle_val_present_flag )
    PicOrderCntMsb = poc_msb_cycle_val * MaxPicOrderCntLsb
else if( !FirstPicInLayerDecodedFlag[ nuh_layer_id ] | |
        nal_unit_type = = IDR_N_LP | | nal_unit_type = = IDR_W_RADL )
    PicOrderCntMsb = 0                                                          (F-62)
else {
    prevPicOrderCntLsb = PrevPicOrderCnt[ nuh_layer_id ] & ( MaxPicOrderCntLsb − 1 )
    prevPicOrderCntMsb = PrevPicOrderCnt[ nuh_layer_id ] − prevPicOrderCntLsb
    PicOrderCntMsb = GetCurrMsb( slice_pic_order_cnt_lsb, prevPicOrderCntLsb,
            prevPicOrderCntMsb, MaxPicOrderCntLsb )
}
PicOrderCntVal = PicOrderCntMsb + slice_pic_order_cnt_lsb
```

The value of PrevPicOrderCnt[lId] for each of the lId values included in affectedLayerList is derived as follows:

– If the current picture is not a RASL, RADL or SLNR picture, and the current picture has TemporalId equal to 0 and discardable_flag equal to 0, PrevPicOrderCnt[lId] is set equal to PicOrderCntVal.

– Otherwise, when poc_reset_idc is equal to 3 and one of the following conditions is true, PrevPicOrderCnt[lId] is set equal to (full_poc_reset_flag ? 0 : poc_lsb_val):

  – FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 0.

  – FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 1 and the current picture is a POC resetting picture.

The value of PicOrderCntVal shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

It is a requirement of bitstream conformance that the current PicOrderCntVal values of any two pictures in the same layer and the same CVS shall not be the same.

NOTE 1 – PicOrderCntVal, as derived in this clause for the current picture, may be equal to initialPocVal, where initialPocVal is the PicOrderCntVal derived by this clause when an earlier picture, in decoding order, with nuh_layer_id equal to currNuhLayerId in the current CVS, was the current picture. However, the decoding processes applied subsequently have updated the PicOrderCntVal value for that earlier picture so that it no longer is equal to the PicOrderCntVal value derived in this clause for the current picture.

The function PicOrderCnt(picX) is specified as follows:

PicOrderCnt(picX) = PicOrderCntVal of the picture picX (F-63)

The function DiffPicOrderCnt(picA, picB) is specified as follows:

DiffPicOrderCnt(picA, picB) = PicOrderCnt(picA) − PicOrderCnt(picB) (F-64)

The bitstream shall not contain data that result in values of DiffPicOrderCnt(picA, picB) used in the decoding process that are not in the range of -2^{15} to $2^{15} - 1$, inclusive.

NOTE 2 – Let X be the current picture and Y and Z be two other pictures in the same sequence, Y and Z are considered to be in the same output order direction from X when both DiffPicOrderCnt(X, Y) and DiffPicOrderCnt(X, Z) are positive or both are negative.

F.8.3.2 Decoding process for reference picture set

The specifications in clause 8.3.2 apply with the following changes:

– The references to clauses 7.4.7.2, 8.3.1, 8.3.3 and 8.3.4 are replaced with references to clauses F.7.4.7.2, F.8.3.1, F.8.3.3 and F.8.3.4, respectively.

– The following specifications are added:

  - When the current picture is an IRAP picture with nuh_layer_id equal to SmallestLayerId, all reference pictures with any value of nuh_layer_id currently in the DPB (if any) are marked as "unused for reference" when at least one of the following conditions is true:
    - The current picture has NoClrasOutputFlag is equal to 1.
    - The current picture activates a new VPS.
- It is a requirement of bitstream conformance that the RPS is restricted as follows:
    - When the current picture is a CRA picture, there shall be no picture in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr.
- The constraints specified in clause 8.3.2 on the value of NumPicTotalCurr are replaced with the following:
  - It is a requirement of bitstream conformance that the following applies to the value of NumPicTotalCurr:
    - If the current picture is a BLA or CRA picture and either currPicLayerId is equal to 0 or NumDirectRefLayers[currPicLayerId] is equal to 0, the value of NumPicTotalCurr shall be equal to pps_curr_pic_ref_enabled_flag.
    - Otherwise, when the current picture contains a P or B slice, the value of NumPicTotalCurr shall not be equal to 0.
- The constraint, specified in clause 8.3.2, that requires no entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr when one or more of three conditions are true is replaced with the following:
    - There shall be no entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr for which one or more of the following are true:
      - The entry is equal to "no reference picture" and FirstPicInLayerDecodedFlag[currPicLayerId] is equal to 1.
      - The entry is an SLNR picture and has TemporalId equal to that of the current picture.
    - The entry is a picture that has TemporalId greater than that of the current picture.

F.8.3.3 Decoding process for generating unavailable reference pictures

The specifications in clause 8.3.3 and its subclauses apply with the replacement of references to Annex C with references to clause F.13.

F.8.3.4 Decoding process for reference picture lists construction

This process is invoked at the beginning of the decoding process for each P or B slice.

Reference pictures are addressed through reference indices as specified in clause 8.5.3.3.2. A reference index is an index into a reference picture list. When decoding a P slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.

At the beginning of the decoding process for each slice, the reference picture lists RefPicList0 and, for B slices, RefPicList1 are derived as follows:

If TwoVersionsOfCurrDecPicFlag is equal to 1, let the variable currPic be the current decoded picture before the invocation of the in-loop filter process; otherwise (TwoVersionsOfCurrDecPicFlag is equal to 0), let the variable currPic be the current decoded picture after the invocation of the in-loop filter process. The variable NumRpsCurrTempList0 is set equal to Max(num_ref_idx_l0_active_minus1 + 1, NumPicTotalCurr) and the list RefPicListTemp0 is constructed as follows:

```
rIdx = 0
while( rIdx < NumRpsCurrTempList0 ) {
    for( i = 0; i < NumPocStCurrBefore  &&  rIdx < NumRpsCurrTempList0; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetStCurrBefore[ i ]
    for( i = 0; i < NumActiveRefLayerPics0; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetInterLayer0[ i ]
    for( i = 0;  i < NumPocStCurrAfter  &&  rIdx < NumRpsCurrTempList0; rIdx++, i++ )          (F-65)
        RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]
    for( i = 0; i < NumPocLtCurr  &&  rIdx < NumRpsCurrTempList0; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]
    for( i = 0; i < NumActiveRefLayerPics1; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetInterLayer1[ i ]
    if( pps_curr_pic_ref_enabled_flag )
```

```
        RefPicListTemp0[ rIdx++ ] = currPic
}
```

The list RefPicList0 is constructed as follows:

```
for( rIdx = 0; rIdx <= num_ref_idx_l0_active_minus1; rIdx++ )
    RefPicList0[ rIdx ] = ref_pic_list_modification_flag_l0 ? RefPicListTemp0[ list_entry_l0[ rIdx ] ] :
                                                                RefPicListTemp0[ rIdx ]
if( pps_curr_pic_ref_enabled_flag && !ref_pic_list_modification_flag_l0 &&              (F-66)
                        NumRpsCurrTempList0 > ( num_ref_idx_l0_active_minus1 + 1 ) )
    RefPicList0[ num_ref_idx_l0_active_minus1 ] = currPic
```

When the slice is a B slice, the variable NumRpsCurrTempList1 is set equal to Max(num_ref_idx_l1_active_minus1 + 1, NumPicTotalCurr) and the list RefPicListTemp1 is constructed as follows:

```
rIdx = 0
while( rIdx < NumRpsCurrTempList1 ) {
    for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrAfter[ i ]
    for( i = 0; i< NumActiveRefLayerPics1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetInterLayer1[ i ]
    for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ )     (F-67)
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]
    for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]
    for( i = 0; i< NumActiveRefLayerPics0; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetInterLayer0[ i ]
    if( pps_curr_pic_ref_enabled_flag )
        RefPicListTemp1[ rIdx++ ] = currPic
}
```

When the slice is a B slice, the list RefPicList1 is constructed as follows:

```
for( rIdx = 0; rIdx <= num_ref_idx_l1_active_minus1; rIdx++ )
    (F-68)
    RefPicList1[ rIdx ] = ref_pic_list_modification_flag_l1 ? RefPicListTemp1[ list_entry_l1[ rIdx ] ] :
                                                                RefPicListTemp1[ rIdx ]
```

It is a requirement of bitstream conformance that when the current layer is an independent non-base layer, nal_unit_type has a value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive (i.e. the picture is an IRAP picture), pps_curr_pic_ref_enabled_flag is equal to 1, and slice_type is not equal to 2, RefPicList0 and RefPicList1 shall not contain entries that refer to a picture other than the current picture.

F.8.4 Decoding process for coding units coded in intra prediction mode

The specifications in clause 8.4 and its subclauses apply.

F.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in clause 8.5 and its subclauses apply.

F.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause 8.6 and its subclauses apply.

F.8.7 In-loop filter process

The specifications in clause 8.7 and its subclauses apply.

F.9 Parsing process

The specifications in clause 9 and its subclauses apply.

F.10 Specification of bitstream subsets

F.10.1 Sub-bitstream extraction process

Inputs to this process are a bitstream inBitstream, a target highest TemporalId value tIdTarget and a target layer identifier list layerIdListTarget.

Output of this process is a sub-bitstream outBitstream.

When vps_base_layer_internal_flag is equal to 1 and vps_base_layer_available_flag is equal to 1, it is a requirement of bitstream conformance for inBitstream that any output sub-bitstream outBitstream that is the output of the process specified in this clause with inBitstream, tIdTarget equal to any value in the range of 0 to 6, inclusive, and layerIdListTarget equal to the layer identifier list associated with a layer set specified in the active VPS as inputs, and that satisfies both of the following conditions shall be a conforming bitstream:

– The output sub-bitstream contains at least one VCL NAL unit with nuh_layer_id equal to each of the nuh_layer_id values in layerIdListTarget.

– The output sub-bitstream contains at least one VCL NAL unit with TemporalId equal to tIdTarget.

NOTE 1 – When vps_base_layer_internal_flag is equal to 1 and vps_base_layer_available_flag is equal to 1, a conforming bitstream contains one or more coded slice segment NAL units with nuh_layer_id equal to 0 and TemporalId equal to 0.

The output sub-bitstream outBitstream is derived as follows:

– The bitstream outBitstream is set to be identical to the bitstream inBitstream.

– When one or more of the following two conditions are true, remove from outBitstream all SEI NAL units that have nuh_layer_id equal to 0 and that contain a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, or a non-scalable-nested decoding unit information SEI message:

  – layerIdListTarget does not include all the values of nuh_layer_id in all NAL units in the bitstream.

  – tIdTarget is less than the greatest TemporalId in all NAL units in the bitstream.

  NOTE 2 – A "smart" bitstream extractor may include appropriate non-scalable-nested buffering picture SEI messages, non-scalable-nested picture timing SEI messages and non-scalable-nested decoding unit information SEI messages in the extracted sub-bitstream, provided that the SEI messages applicable to the sub-bitstream were present as scalable-nested SEI messages in the original bitstream.

– Remove from outBitstream all NAL units with TemporalId greater than tIdTarget or nuh_layer_id not among the values included in layerIdListTarget.

F.10.2 Independent non-base layer rewriting process

Inputs to this process are a bitstream inBitstream, a target highest TemporalId value tIdTarget and a nuh_layer_id value assignedBaseLayerId of an independent non-base layer of an additional layer set.

Output of this process is a sub-bitstream outBitstream.

It is a requirement of bitstream conformance for inBitstream that any output sub-bitstream outBitstream that is the output of the process specified in this clause shall otherwise be a conforming bitstream except that outBitstream does not contain any VPS NAL units, when all of the following conditions apply:

– The output sub-bitstream outBitstream contains VCL NAL units with nuh_layer_id equal to assignedBaseLayerId.

– The value of tIdTarget is equal to any value in the range of 0 to 6, inclusive, and outBitstream contains at least one VCL NAL unit with TemporalId equal to tIdTarget.

– There is an OLS in the active VPS that consists of only the layer with nuh_layer_id equal to assignedBaseLayerId, the profile of that layer is a profile specified in Annex A and the value of general_inbld_flag (when tIdTarget is equal to vps_max_sub_layers_minus1) or the value of sub_layer_inbld_flag[tidTarget] (when tidTarget is less than vps_max_sub_layers_minus1) in the profile_tier_level( ) syntax structure associated with that layer is equal to 1.

The output sub-bitstream outBitstream is derived from the bitstream inBitstream as follows:

– The bitstream outBitstream is set to be identical to the bitstream inBitstream.

– NAL units with nal_unit_type not equal to SPS_NUT, PPS_NUT, and EOB_NUT and with nuh_layer_id not equal to the assignedBaseLayerId are removed from outBitstream.

– NAL units with nal_unit_type equal to SPS_NUT or PPS_NUT with nuh_layer_id not equal to 0 or assignedBaseLayerId are removed from outBitstream.

– NAL units with nal_unit_type equal to VPS_NUT are removed from outBitstream.

– All NAL units with TemporalId greater than tIdTarget are removed from outBitstream.

– nuh_layer_id is set equal to 0 in each NAL unit of outBitstream.

F.10.3 Sub-bitstream extraction process for additional layer sets

Inputs to this process are a bitstream inBitstream, a target highest TemporalId value tIdTarget and a target layer identifier list layerIdListTarget of an additional layer set.

Output of this process is a sub-bitstream outBitstream.

It is a requirement of bitstream conformance for the input bitstream that the output sub-bitstream of the process specified in this clause shall be a conforming bitstream according to at least one profile in which vps_base_layer_available_flag may be equal to 0, when all of the following conditions apply:

– The output sub-bitstream outBitstream contains VCL NAL units with each nuh_layer_id value in layerIdListTarget.

– The value of tIdTarget is equal to any value in the range of 0 to 6, inclusive, and outBitstream contains at least one VCL NAL unit with TemporalId equal to tIdTarget.

– layerIdListTarget is identical to LayerSetLayerIdList[i] for any value of i in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive.

The output sub-bitstream outBitstream is derived as follows:

– The bitstream outBitstream is set to be identical to the bitstream inBitstream.

– NAL units with nal_unit_type not equal to VPS_NUT, SPS_NUT, PPS_NUT, EOS_NUT and EOB_NUT and with nuh_layer_id not equal to any value in layerIdListTarget are removed from outBitstream.

– NAL units with nal_unit_type equal to VPS_NUT, SPS_NUT, PPS_NUT or EOS_NUT with nuh_layer_id not equal to 0 or any value in layerIdListTarget are removed from outBitstream.

– All NAL units with TemporalId greater than tIdTarget are removed from outBitstream.

– vps_base_layer_available_flag in each VPS is set equal to 0.

F.11 Profiles, tiers and levels

F.11.1 Independent non-base layer decoding capability

This clause specifies the independent non-base layer decoding (INBLD) capability, which is associated with the decoding capability of one or more of the profiles specified in Annex A. When expressing the capabilities of a decoder for one or more profiles specified in Annex A, whether the INBLD capability is supported for those profiles should also be expressed.

NOTE 1– The INBLD capability, when supported, indicates the capability of a decoder to decode an independent non-base layer that is indicated in the active VPSs and SPSs to conform to a profile specified in Annex A and is the layer with the smallest nuh_layer_id value in an additional layer set.

When the profile_tier_level( ) syntax structure is used for indicating of a decoder capability in systems, the INBLD capability may be indicated by setting the general_inbld_flag equal to 1 in the profile_tier_level( ) syntax structure used to express the profile, tier and level that the decoder conforms to.

general_inbld_flag is set equal to 1 in the profile_tier_level( ) syntax structures in which a profile specified in Annex A is indicated and which are either specified in the VPS to be applicable for a non-base layer or included in an SPS activated for an independent non-base layer.

Decoders having the INBLD capability and conforming to a specific profile specified in Annex A at a specific level of a specific tier shall be capable of decoding any independent non-base layer or a sub-layer representation with TemporalId equal to i of the independent non-base layer for which all of the following condition applies for each active VPS:

– There is an OLS that consists of the independent non-base layer and for which the associated profile_tier_level( ) syntax structure ptlStruct is constrained as follows:

  – ptlStruct indicates that the independent non-base layer or the sub-layer representation conforms to a profile specified in Annex A.

  – ptlStruct indicates that the independent non-base layer or the sub-layer representation conforms to a level lower than or equal to the specified level.

  – ptlStruct indicates that the independent non-base layer or the sub-layer representation conforms to a tier lower than or equal to the specified tier.

  – general_inbld_flag or sub_layer_inbld_flag[i] in ptlStruct is equal to 1.

NOTE 2– Let a derived bitstream outBitstream be a bitstream which is derived by invoking the independent non-base layer rewriting process specified in clause F.10.2 with a bitstream containing one or more independent non-base layers, tIdTarget equal to 6 and assignedBaseLayerId equal to the smallest nuh_layer_id value of additional layer set as inputs. As specified elsewhere in this Specification, it is a requirement of bitstream conformance that outBitstream is otherwise a conforming bitstream but does not contain VPSs. Consequently, decoders with INBLD capability may apply the independent non-base layer rewriting process specified in clause F.10.2 to obtain outBitstream and then apply a decoding process for a profile specified in Annex A with outBitstream as input.

NOTE 3– The following constraints are necessary to ensure that a sub-bitstream derived by invoking the independent non-base layer rewriting process specified in clause F.10.2 conforms to a profile specified in Annex A:

– All active SPSs for the independent non-base layer have nuh_layer_id equal to 0 or MultiLayerExtSpsFlag equal to 0.

– Each active VPS has poc_lsb_not_present_flag[i] equal to 1 for each value of i for which i is the layer index of the independent non-base layer.

F.11.2 Decoder capabilities

This clause specifies requirements for decoders having the capability of decoding an output operation point with one or more necessary layers.

NOTE – For example, this clause specifies the requirements for a decoder supporting simultaneous decoding of the base layer and an independent non-base layer for which the INBLD capability is needed. Moreover, this clause specifies the requirements for a decoder supporting decoding of layers conforming to profiles specified in Annex G or H.

A decoder that conforms to a list ptliList consisting of profile, tier, level and INBLD capability quadruplets (P_i, T_i, L_i, $InbldFlag_i$) for i in the range of 0 to d − 1, inclusive, where d is a positive integer, shall be capable of decoding any output operation point containing b necessary layers, where b is less than or equal to d, when the following condition applies:

– There exists a reordered list, orderedPtliList, of the profile, tier, level and INBLD capability quadruplets ($orderedP_j$, $orderedT_j$, $orderedL_j$, $orderedInbldFlag_j$) of the list ptliList such that all the following conditions apply for each value of j in the range of 0 to b − 1, inclusive, where the variable bLayer[j] represents the j-th necessary layer in the output operation point:

  – The profile to which bLayer[j] is indicated to conform is included in CompatibleProfileList for the profile $orderedP_j$ as specified in Table F.3.

  – The tier to which bLayer[j] is indicated to conform is lower than or equal to the tier $orderedT_j$.

  – The level to which bLayer[j] is indicated to conform is lower than or equal to the level $orderedL_j$.

  – When the profile to which bLayer[j] is indicated to conform is a profile specified in Annex A, the value of general_inbld_flag (when OpTid of the output operation point is equal to vps_max_sub_layers_minus1) or sub_layer_inbld_flag[OpTid] (when OpTid of the output operation point is less than vps_max_sub_layers_minus1) of bLayer[j] is less than or equal to $orderedInbldFlag_j$.

For each particular format range extensions profile specified in clause A.3.5, the compatible format range extensions profiles are defined as the profiles that are indicated by both of the following conditions being true:

– The value of general_profile_idc is equal to 4 or general_profile_compatibility_flag[4] is equal to 1.

– The value of each constraint flag listed in Table A.2 is less than or equal to the corresponding value specified in the row of Table A.2 for the particular format range extensions profile.

For each particular scalable format range extensions profile specified in clause H.11.1.2, the compatible scalable format range extensions profiles are defined as the profiles that are indicated by all of the following conditions being true:

– The value of general_profile_idc is equal to 10 or general_profile_compatibility_flag[10] is equal to 1.

– The value of each constraint flag listed in Table H.4 is less than or equal to the corresponding value specified in the row of Table H.4 for the particular scalable format range extensions profile.

Table F.3 – Specification of CompatibleProfileList

Profile to which the decoder conforms	Profiles that the decoder shall support CompatibleProfileList
Scalable Main	Scalable Main, Main, Main Still Picture
Scalable Main 10	Scalable Main 10, Main, Main Still Picture, Main 10, Scalable Main
Scalable Monochrome	The compatible format range extensions profiles of the Monochrome profile, and the compatible scalable format range extensions profiles of the Scalable Monochrome profile
Scalable Monochrome 12	The compatible format range extensions profiles of the Monochrome 12 profile, and the compatible scalable format range extensions profiles of the Scalable Monochrome 12 profile
Scalable Monochrome 16	The compatible format range extensions profiles of the Monochrome 16 profile, and the compatible scalable format range extensions profiles of the Scalable Monochrome 16 profile
Scalable Main 4:4:4	Scalable Main, Main, Main Still Picture, the compatible format range extensions profiles of the Main 4:4:4 profile, and the compatible scalable format range extensions profiles of the Scalable Main 4:4:4 profile
Multiview Main	Multiview Main, Main, Main Still Picture
3D Main	3D Main, Multiview Main, Main, Main Still Picture

F.11.3 Derivation of sub-bitstreams subBitstream and baseBitstream

For an output operation point associated with an OLS in a bitstream, let olsIdx be the OLS index of the OLS, the sub-bitstream subBitstream and base layer sub-bitstream baseBitstream are derived as follows:

– The sub-bitstream subBitstream is derived as follows:

  – If OlsIdxToLsIdx[olsIdx] is less than or equal to vps_num_layer_sets_minus1, subBitstream is derived by invoking the sub-bitstream extraction process as specified in clause F.10.1 with the following inputs: the bitstream, tIdTarget equal to OpTid of the output operation point, and layerIdListTarget containing the nuh_layer_id value layerId of the layer and all the reference layers of the layer.

  – Otherwise, subBitstream is derived by invoking the sub-bitstream extraction process as specified in clause F.10.3 with tIdTarget equal to OpTid of the output operation point and with layerIdListTarget containing the nuh_layer_id value of the layer and all the reference layers of the layer.

– When vps_base_layer_internal_flag is equal to 1, the base layer sub-bitstream baseBitstream is derived as follows:

  – If VCL NAL units with nuh_layer_id equal to 0 are included in subBitstream, baseBitstream is derived by invoking the sub-bitstream extraction process as specified in clause F.10.1 with the subBitstream, tIdTarget equal to OpTid of the output operation point, and layerIdListTarget containing only one nuh_layer_id value that is equal to 0 as inputs.

  – Otherwise, baseBitstream is derived by invoking the independent non-base layer rewriting process as specified in clause F.10.2 with subBitstream, tIdTarget equal to OpTid of the output operation point, and layerIdListTarget containing only the smallest nuh_layer_id value of the VCL NAL units of subBitstream as inputs.

F.12 Byte stream format

The specifications in Annex B apply.

F.13 Hypothetical reference decoder

F.13.1 General

Three sets of bitstream conformance tests are needed for checking the conformance of a bitstream, which is referred to as the entire bitstream, denoted as entireBitstream. The first set of bitstream conformance tests are for testing the conformance of the entire bitstream and its temporal subsets, regardless of whether there is a layer set specified by the active VPS that contains all the nuh_layer_id values of VCL NAL units present in the entire bitstream. The second set of bitstream

conformance tests are for testing the conformance of the layer sets specified by the active VPS and their temporal subsets. For all these tests, only the base layer pictures (i.e., pictures with nuh_layer_id equal to 0) are decoded and other pictures are ignored by the decoder when the decoding process is invoked.

The first and second sets of bitstream conformance tests are specified in clause C.1. Clause F.13 and its subclauses specify the HRD operations for the third sets of bitstream conformance tests.

The third set of bitstream conformance tests are for testing the conformance of the OLSs specified by the VPS extension part of the active VPS and their temporal subsets. For each test in the third set of bitstream conformance tests, the following ordered steps apply in the order listed, followed by the processes described after these steps in this clause:

1. An output operation point under test, denoted as TargetOp, is selected by selecting a value for TargetOlsIdx identifying a target OLS and selecting a target highest TemporalId value HighestTid. The value of TargetOlsIdx shall be in the range of 0 to NumOutputLayerSets − 1, inclusive, and the value of HighestTid shall be in the range of 0 to MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[TargetOlsIdx]], inclusive. Additionally, the values of TargetOlsIdx and HighestTid are constrained as specified in the next step.

   The variables TargetDecLayerSetIdx, TargetOptLayerIdList, and TargetDecLayerIdList are then derived as specified in clause F.8.1.2. The output operation point under test has OptLayerIdList equal to TargetOptLayerIdList, OpLayerIdList equal to TargetDecLayerIdList and OpTid equal to HighestTid.

2. A bitstream to be decoded, BitstreamToDecode, is specified as follows:
   - If TargetDecLayerSetIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 1, the sub-bitstream extraction process as specified in clause 10 is applied with the bitstream entireBitstream, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to BitstreamToDecode.
   - Otherwise, if TargetDecLayerSetIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 0, the sub-bitstream extraction process as specified in clause F.10.1 is applied with the bitstream entireBitstream, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to BitstreamToDecode.
   - Otherwise, if TargetDecLayerSetIdx is greater than vps_num_layer_sets_minus1, NumLayersInIdList[TargetDecLayerSetIdx] is equal to 1 and the profile of the layer in TargetOlsIdx is one of those specified in Annex A, the independent non-base layer rewriting process of clause F.10.2 is applied with the bitstream entireBitstream, HighestTid and TargetDecLayerIdList[0] as inputs, and the output is assigned to BitstreamToDecode.
   - Otherwise, the sub-bitstream extraction process as specified in clause F.10.3 is applied with the bitstream entireBitstream, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to BitstreamToDecode.

   The values of TargetOlsIdx and HighestTid are additionally constrained as follows:
   - When vps_base_layer_available_flag is equal to 0, OlsIdxToLsIdx[TargetOlsIdx] shall be in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive.
   - When vps_base_layer_internal_flag is equal to 0, TargetOlsIdx shall be greater than 0.
   - The value of TargetOlsIdx shall be such that there is at least one VCL NAL unit in BitstreamToDecode with nuh_layer_id equal to LayerSetLayerIdList[OlsIdxToLsIdx[TargetOlsIdx]][i] for each value of i in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[TargetOlsIdx]] − 1, inclusive.
   - The value of HighestTid shall be such that there is at least one VCL NAL unit with TemporalId equal to HighestTid in BitstreamToDecode.

3. A partitioning scheme is selected from the list of partitioning schemes signalled in the active VPS for the selected OLS. The selected partitioning scheme is denoted as TargetPartitioningScheme with partitioning scheme index TargetPsIdx.

4. The subsequent steps apply to each bitstream partition, referred to as the bitstream partition under test TargetBitstreamPartition, of the selected partitioning scheme of the target OLS. If there is only one bitstream partition for TargetPartitioningScheme, the TargetBitstreamPartition is identical to BitstreamToDecode. Otherwise, each bitstream partition is derived with the demultiplexing process for deriving a bitstream partition in clause F.13.6, with BitstreamToDecode, the list of layers in TargetBitstreamPartition and the number of layers in TargetBitstreamPartition as inputs.

5. The applicable hrd_parameters( ) syntax structures and the sub_layer_hrd_parameters( ) syntax structures are selected as follows:
   - A SchedSelCombIdx is selected for BitstreamToDecode and used for each TargetBitstreamPartition. The selected SchedSelCombIdx shall be in the range of 0 to num_bsp_schedules_minus1[TargetOlsIdx][TargetPsIdx][HighestTid], inclusive.

– The applicable hrd_parameters( ) syntax structure is the bsp_hrd_idx[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j]-th hrd_parameters( ) syntax structure in the active VPS, where j is the bitstream partition index of TargetBitstreamPartition.

Within the selected hrd_parameters( ) syntax structures, if BitstreamToDecode is a Type I bitstream, the sub_layer_hrd_parameters(HighestTid) syntax structures that immediately follow the condition "if(vcl_hrd_parameters_present_flag)" are selected and the variable NalHrdModeFlag is set equal to 0; otherwise (BitstreamToDecode is a Type II bitstream), the sub_layer_hrd_parameters(HighestTid) syntax structures that immediately follow either the condition "if(vcl_hrd_parameters_present_flag)" (in this case the variable NalHrdModeFlag is set equal to 0) or the condition "if(nal_hrd_parameters_present_flag)" (in this case the variable NalHrdModeFlag is set equal to 1) are selected. When BitstreamToDecode is a Type II bitstream and NalHrdModeFlag is equal to 0, all non-VCL NAL units except filler data NAL units, and all leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes, and trailing_zero_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B), when present, are discarded from TargetBitstreamPartition, and the remaining sub-bitstream is assigned to TargetBitstreamPartition.

6. An access unit associated with a buffering period SEI message (present in a bitstream partition nesting SEI message in BitstreamToDecode or available through external means not specified in this Specification) applicable to TargetOp, TargetPartitioningScheme and TargetBitstreamPartition is selected as the HRD initialization point and referred to as access unit 0.

   The variable MultiLayerCpbOperationFlag is derived as follows:

   – If BitstreamToDecode contains only the base layer (i.e., TargetOlsIdx is equal to 0), MultiLayerCpbOperationFlag is set equal to 0.

   – Otherwise, MultiLayerCpbOperationFlag is set equal to 1.

7. If sub_pic_hrd_params_present_flag in the selected hrd_parameters( ) syntax structure is equal to 1, the CPB is scheduled to operate either at the partition unit level (in which case the variable SubPicHrdFlag is set equal to 0) or at the sub-partition level (in which case the variable SubPicHrdFlag is set equal to 1). Otherwise, SubPicHrdFlag is set equal to 0 and the CPB is scheduled to operate at the partition unit level.

8. For each access unit in TargetBitstreamPartition starting from access unit 0, the buffering period SEI message (present in a bitstream partition nesting SEI message in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp, TargetPartitioningScheme and TargetBitstreamPartition is selected, the picture timing SEI message (present in a bitstream partition nesting SEI message in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp, TargetPartitioningScheme and TargetBitstreamPartition is selected, and when SubPicHrdFlag is equal to 1 and sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 0, the decoding unit information SEI messages (present in bitstream partition nesting SEI messages in BitstreamToDecode or available through external means not specified in this Specification) that are associated with decoding units in the access unit and apply to TargetOp, TargetPartitioningScheme, and TargetBitstreamPartition are selected.

9. A value of SchedSelIdx for TargetBitstreamPartition is set equal to bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j], where j is the index of the bitstream partition index of TargetBitstreamPartition.

10. The variable initialAltParamSelectionFlag is derived as follows:

    – If all of the following conditions are true, initialAltParamSelectionFlag is set equal to 1:

      – The coded picture with nuh_layer_id equal to SmallestLayerId in access unit 0 has nal_unit_type equal to CRA_NUT or BLA_W_LP.

      – MultiLayerCpbOperationFlag is equal to 0.

      – irap_cpb_params_present_flag in the selected buffering period SEI message is equal to 1.

    – Otherwise, if all of the following conditions are true, initialAltParamSelectionFlag is set equal to 1:

      – The coded picture with nuh_layer_id equal to SmallestLayerId in access unit 0 is an IRAP picture.

      – MultiLayerCpbOperationFlag is equal to 1.

      – irap_cpb_params_present_flag in the selected buffering period SEI message is equal to 1.

    – Otherwise, initialAltParamSelectionFlag is set equal to 0.

11. When initialAltParamSelectionFlag is equal to 1, the following applies:

    – The variable skippedPictureList is set to consist of the CL-RAS pictures and the RASL pictures associated with the IRAP pictures with nuh_layer_id equal to nuhLayerId for which LayerInitializedFlag[nuhLayerId] is equal to 0 at the start of decoding the IRAP picture and for which nuhLayerId is among TargetDecLayerIdList.

- Either of the following applies for selection of the initial CPB removal delay and delay offset:
  - If NalHrdModeFlag is equal to 1, the default initial CPB removal delay and delay offset represented by nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. Otherwise, the default initial CPB removal delay and delay offset represented by vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. The variable DefaultInitCpbParamsFlag is set equal to 1.
  - If NalHrdModeFlag is equal to 1, the alternative initial CPB removal delay and delay offset represented by nal_initial_alt_cpb_removal_delay[SchedSelIdx] and nal_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. Otherwise, the alternative initial CPB removal delay and delay offset represented by vcl_initial_alt_cpb_removal_delay[SchedSelIdx] and vcl_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. The variable DefaultInitCpbParamsFlag is set equal to 0 and all the pictures in skippedPictureList are discarded from BitstreamToDecode and the remaining bitstream is assigned to BitstreamToDecode.

Each conformance test consists of a combination of one option in each of the above steps. When there is more than one option for a step, for any particular conformance test only one option is chosen. All possible combinations of all the steps form the entire set of conformance tests.

When BitstreamToDecode is a Type II bitstream, the following applies:

- If the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows the condition "if(vcl_hrd_parameters_present_flag)" is selected, the test is conducted at the Type I conformance point and only VCL and filler data NAL units are counted for the input bit rate and CPB storage.
- Otherwise (the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows the condition "if(nal_hrd_parameters_present_flag)" is selected), the test is conducted at the Type II conformance point, and all bytes of the Type II bitstream, which may be a NAL unit stream or a byte stream, are counted for the input bit rate and CPB storage.

  NOTE 1 – NAL HRD parameters established by a value of SchedSelIdx for the Type II conformance point are sufficient to also establish VCL HRD conformance for the Type I conformance point for the same values of InitCpbRemovalDelay[SchedSelIdx], BitRate[SchedSelIdx] and CpbSize[SchedSelIdx] for the VBR case (cbr_flag[SchedSelIdx] equal to 0). This is because the data flow into the Type I conformance point is a subset of the data flow into the Type II conformance point and because, for the VBR case, the CPB is allowed to become empty and stay empty until the time a next picture is scheduled to begin to arrive. For example, when decoding a CVS conforming to one or more of the profiles specified in Annex G using the decoding process specified in clauses G.2 through G.10, when NAL HRD parameters are provided for the Type II conformance point that not only fall within the bounds set for NAL HRD parameters for profile conformance in item f) of clause G.11.2.2 but also fall within the bounds set for VCL HRD parameters for profile conformance in item e) of clause G.11.2.2, conformance of the VCL HRD for the Type I conformance point is also assured to fall within the bounds of item e) of clause G.11.2.2.

All VPSs, SPSs and PPSs referred to in the VCL NAL units, and the corresponding buffering period, picture timing and decoding unit information SEI messages shall be conveyed to the HRD, in a timely manner, either in the bitstream (by non-VCL NAL units), or by other means not specified in this Specification.

In Annexes C, D and E, the specification for "presence" of non-VCL NAL units that contain VPSs, SPSs, PPSs, buffering period SEI messages, picture timing SEI messages or decoding unit information SEI messages is also satisfied when those NAL units (or just some of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

NOTE 2 – As an example, synchronization of such a non-VCL NAL unit, conveyed by means other than presence in the bitstream, with the NAL units that are present in the bitstream, can be achieved by indicating two points in the bitstream, between which the non-VCL NAL unit would have been present in the bitstream, had the encoder decided to convey it in the bitstream.

When the content of such a non-VCL NAL unit is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the non-VCL NAL unit is not required to use the same syntax as specified in this Specification.

NOTE 3 – When HRD information is contained within the bitstream, it is possible to verify the conformance of a bitstream to the requirements of this clause based solely on information contained in the bitstream. When the HRD information is not present in the bitstream, as is the case for all "stand-alone" Type I bitstreams, conformance can only be verified when the HRD data are supplied by some other means not specified in this Specification.

For the bitstream-partition-specific CPB operation as specified in this annex, the HRD contains a bitstream demultiplexer (optionally present), two or more bitstream partition buffers (BPB), two or more instantaneous decoding processes, a decoded picture buffer (DPB) that contains a sub-DPB for each layer, and output cropping as shown in Figure F.1.

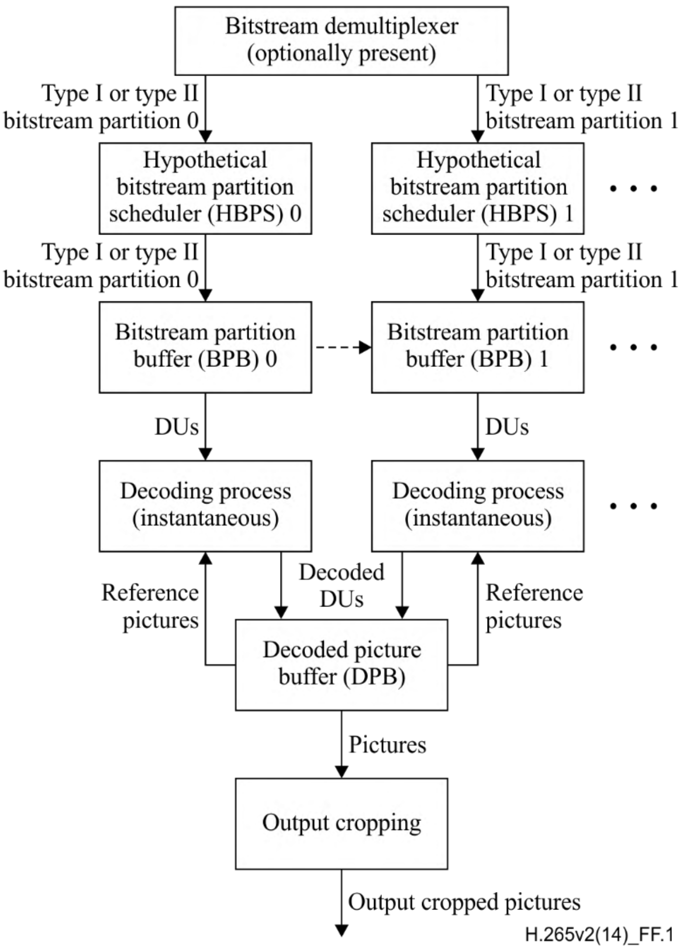


Figure F.1 – Bitstream-partition-specific HRD buffer model

For each bitstream conformance test, the size of BPB (or CPB) CpbSize[SchedSelIdx] is BpbSize[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j], and the BitRate[SchedSelIdx] is BpBitRate[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j], where j is the bitstream partition index of TargetBitstreamPartition, and SchedSelIdx and the HRD parameters are specified above in this clause. The sub-DPB size of the sub-DPB for a layer with nuh_layer_id equal to currLayerId is max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid] + 1, where layerIdx is equal to the value such that LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx] is equal to currLayerId.

If SubPicHrdFlag is equal to 0, the HRD operates at partition unit level and each decoding unit is a partition unit. Otherwise the HRD operates at sub-partition level and each decoding unit is a subset of a partition unit.

NOTE 4 – If the HRD operates at partition unit level, each time when some bits are removed from the CPB, a decoding unit that is an entire partition unit is removed from the CPB. Otherwise (the HRD operates at sub-partition level), each time when some bits are removed from the CPB, a decoding unit that is a subset of a partition unit is removed from the CPB. Regardless of whether the HRD operates at partition unit level or sub-partition level, each time when some picture is output from the DPB, an entire decoded picture is output from the DPB, though the picture output time is derived based on the differently derived CPB removal times and the differently signalled DPB output delays.

The following is specified for expressing the constraints in this annex:

– Each access unit is referred to as access unit n, where the number n identifies the particular access unit. Access unit 0 is selected per step 6 above. The value of n is incremented by 1 for each subsequent access unit in decoding order.

– Each decoding unit is referred to as decoding unit m, where the number m identifies the particular decoding unit. The first decoding unit in decoding order in access unit 0 is referred to as decoding unit 0. The value of m is incremented by 1 for each subsequent decoding unit in decoding order.

  NOTE 5 – The numbering of decoding units is relative to the first decoding unit in access unit 0.

– Picture n refers to a particular coded or decoded picture of access unit n.

The HRD operates as follows:

– The HRD is initialized at decoding unit 0, with the CPB, each sub-DPB of the DPB and each BPB being set to be empty (the sub-DPB fullness for each sub-DPB is set equal to 0).

NOTE 6 – After initialization, the HRD is not initialized again by subsequent buffering period SEI messages.

– Data associated with decoding units that flow into the BPB according to a specified arrival schedule are delivered by an HBPS.

– Each bitstream partition with index j is processed as specified in clause F.13.2 with the HSS replaced by the HBPS and with SchedSelIdx equal to bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j].

– The data associated with each decoding unit are removed and decoded instantaneously by the instantaneous decoding process at the CPB removal time of the decoding unit.

– Each decoded picture is placed in the DPB.

– A decoded picture is removed from the DPB when it becomes no longer needed for inter prediction reference and no longer needed for output.

For each bitstream conformance test, the operation of the CPB and the BPB is specified in clause F.13.2, the instantaneous decoder operation is specified in clauses 2 through 10, clauses F.2 through F.10, and either clauses G.2 through G.10 or clauses H.2 through H.10, the operation of the DPB is specified in clause F.13.3, and the output cropping is specified in clauses F.13.3.3 and F.13.5.2.2.

HSS, HBPS and HRD information concerning the number of enumerated delivery schedules and their associated bit rates and buffer sizes is specified in clauses E.2.2, E.2.3, F.7.3.2.1.6 and F.7.4.3.1.6. The HRD is initialized as specified by the buffering period SEI message specified in clauses D.2.2 and D.3.2. The removal timing of decoding units from the CPB and output timing of decoded pictures from the DPB is specified using information in picture timing SEI messages (specified in clauses D.2.3 and D.3.3) or in decoding unit information SEI messages (specified in clauses D.2.22 and D.3.22). All timing information relating to a specific decoding unit shall arrive prior to the CPB removal time of the decoding unit.

The requirements for bitstream conformance are specified in clause F.13.4 and the HRD is used to check conformance of bitstreams as specified above in this clause and to check conformance of decoders as specified in clause F.13.5.

F.13.2 Operation of bitstream partition buffer

F.13.2.1 General

The specifications in this clause apply independently to each set of bitstream partition buffer (BPB) parameters that is present and to both the Type I and Type II conformance points and the set of BPB parameters is selected as specified in clause F.13.1. In clause F.13.2 and its subclauses, CPB is understood to be BPB and access unit is understood to be partition unit.

F.13.2.2 Timing of decoding unit arrival

The variable altParamSelectionFlag is derived as follows:

– If all of the following conditions are true, altParamSelectionFlag is set equal to 1:
  – The current picture is a BLA picture that has nal_unit_type equal to BLA_W_LP and nuh_layer_id equal to SmallestLayerId or is a CRA picture that has nuh_layer_id equal to SmallestLayerId.
  – MultiLayerCpbOperationFlag is equal to 0.
– Otherwise, if all of the following conditions are true, altParamSelectionFlag is set equal to 1:
  – The current picture is an IRAP picture with nuh_layer_id equal to SmallestLayerId and with NoClrasOutputFlag equal to 1.
  – MultiLayerCpbOperationFlag is equal to 1.
– Otherwise, altParamSelectionFlag is set equal to 0.

When altParamSelectionFlag is equal to 1, the following applies:

  – If some external means not specified in this Specification is available to set the variable UseAltCpbParamsFlag to a value, UseAltCpbParamsFlag is set equal to the value provided by the external means.
  – Otherwise, UseAltCpbParamsFlag is set equal to the value of use_alt_cpb_params_flag of the buffering period SEI message selected as specified in clause F.13.1.

If SubPicHrdFlag is equal to 0, the variable subPicParamsFlag is set equal to 0 and the process specified in the remainder of this clause is invoked with a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n.

Otherwise (SubPicHrdFlag is equal to 1), the process specified in the remainder of this clause is first invoked with the variable subPicParamsFlag set equal to 0 and a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n, and then invoked with subPicParamsFlag set equal to 1 and a decoding unit being considered as a subset of an access unit, for derivation of the initial and final CPB arrival times for the decoding units in access unit n.

The variables InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are derived as follows:

– If one or more of the following conditions are true, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_alt_cpb_removal_delay[SchedSelIdx] and nal_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_alt_cpb_removal_delay[SchedSelIdx] and vcl_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause F.13.1:
  – Access unit 0 includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_RADL or BLA_N_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1.
  – Access unit 0 includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_LP or includes a CRA picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 0, and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:
    – UseAltCpbParamsFlag for access unit 0 is equal to 1.
    – DefaultInitCpbParamsFlag is equal to 0.
  – Access unit 0 includes an IRAP picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 1 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:
    – UseAltCpbParamsFlag for access unit 0 is equal to 1.
    – DefaultInitCpbParamsFlag is equal to 0.
  – The value of subPicParamsFlag is equal to 1.
– Otherwise, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause F.13.1.

The time at which the first bit of decoding unit m begins to enter the CPB is referred to as the initial arrival time initArrivalTime[m].

Decoding units are indexed in decoding order within the bitstream.

The initial arrival time of decoding unit m is derived as follows:

– If the decoding unit is decoding unit 0 (i.e., when m is equal to 0) and the decoding unit belongs to the base bitstream partition, initArrivalTime[0] is set equal to 0.
– Otherwise, if the decoding unit is decoding unit 0 and the decoding unit does not belong to the base bitstream partition, initArrivalTime[0] is obtained from the bitstream partition initial arrival time SEI message that applies to TargetOp, TargetPartitioningScheme, and TargetBitstreamPartition and is associated with the access unit containing decoding unit 0. The applicable bitstream partition initial arrival time SEI message is either included in a bitstream partition nesting SEI message in an SEI NAL unit in the access unit containing decoding unit 0 in BitstreamToDecode or available through external means not specified in this Specification. If NalHrdModeFlag is equal to 0, initArrivalTime[0] is set equal to vcl_initial_arrival_delay[SchedSelIdx] in the applicable bitstream partition initial arrival time SEI message. Otherwise (NalHrdModeFlag is equal to 1), initArrivalTime[0] is set equal to nal_initial_arrival_delay[SchedSelIdx] in the applicable bitstream partition initial arrival time SEI message.
– Otherwise, the following applies:

– If cbr_flag[SchedSelIdx] is equal to 1, the initial arrival time for decoding unit m is equal to the final arrival time (which is derived below) of decoding unit m − 1, i.e.,

```
if( !subPicParamsFlag )
    initArrivalTime[ m ] = AuFinalArrivalTime[ m − 1 ]                                  (F-69)
else
    initArrivalTime[ m ] = DuFinalArrivalTime[ m − 1 ]
```

– Otherwise (cbr_flag[SchedSelIdx] is equal to 0), the initial arrival time for decoding unit m is derived as follows:

```
if( !subPicParamsFlag )
    initArrivalTime[ m ] = Max( AuFinalArrivalTime[ m − 1 ], initArrivalEarliestTime[ m ] )
    (F-70)
else
    initArrivalTime[ m ] = Max( DuFinalArrivalTime[ m − 1 ], initArrivalEarliestTime[ m ] )
```

where initArrivalEarliestTime[m] is derived as follows:

– The variable tmpNominalRemovalTime is derived as follows:

```
if( !subPicParamsFlag )
    tmpNominalRemovalTime = AuNominalRemovalTime[ m ]                                  (F-71)
else
    tmpNominalRemovalTime = DuNominalRemovalTime[ m ]
```

where AuNominalRemovalTime[m] and DuNominalRemovalTime[m] are the nominal CPB removal time of access unit m and decoding unit m, respectively, as specified in clause F.13.2.3.

– If decoding unit m is not the first decoding unit of a subsequent buffering period, initArrivalEarliestTime[m] is derived as follows:

```
initArrivalEarliestTime[ m ] = tmpNominalRemovalTime −
( InitCpbRemovalDelay[ SchedSelIdx ]
        + InitCpbRemovalDelayOffset[ SchedSelIdx ] ) ÷ 90 000                          (F-72)
```

– Otherwise (decoding unit m is the first decoding unit of a subsequent buffering period), initArrivalEarliestTime[m] is derived as follows:

```
initArrivalEarliestTime[ m ] = tmpNominalRemovalTime −
        ( InitCpbRemovalDelay[ SchedSelIdx ] ÷ 90 000 )                                (F-73)
```

The final arrival time for decoding unit m is derived as follows:

```
if( !subPicParamsFlag )
    AuFinalArrivalTime[ m ] = initArrivalTime[ m ] + sizeInbits[ m ] ÷ BitRate[ SchedSelIdx ]
    (F-74)
else
    DuFinalArrivalTime[ m ] = initArrivalTime[ m ] + sizeInbits[ m ] ÷ BitRate[ SchedSelIdx ]
```

where sizeInbits[m] is the size in bits of decoding unit m, counting the bits of the VCL NAL units and the filler data NAL units for the Type I conformance point or all bits of the Type II bitstream for the Type II conformance point.

The values of SchedSelIdx, BitRate[SchedSelIdx] and CpbSize[SchedSelIdx] are constrained as follows:

– If the lists of delivery schedules, including the HRD parameters associated with the individual delivery schedules, for the access unit containing decoding unit m and the previous access unit differ, the HSS selects a value SchedCombIdx1 in the range of 0 to num_bsp_schedules_minus1[TargetOlsIdx][TargetPsIdx][HighestTid], inclusive, and sets SchedSelIdx1 for TargetBitstreamPartition to be equal to bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j], where j is the index of the bitstream partition index of TargetBitstreamPartition, for the access unit containing decoding unit m that results in a BitRate[SchedSelIdx1] or CpbSize[SchedSelIdx1] for the access unit containing decoding unit m. The value of BitRate[SchedSelIdx1] or CpbSize[SchedSelIdx1] may differ from the value of BitRate[SchedSelIdx0] or CpbSize[SchedSelIdx0] for the value SchedSelIdx0 of SchedSelIdx that was in use for the previous access unit.

– Otherwise, the HSS continues to operate with the previous values of SchedSelIdx, BitRate[SchedSelIdx] and CpbSize[SchedSelIdx].

When the HSS selects values of BitRate[SchedSelIdx] or CpbSize[SchedSelIdx] that differ from those of the previous access unit, the following applies:

– The variable BitRate[SchedSelIdx] comes into effect at the initial CPB arrival time of the current access unit.

– The variable CpbSize[SchedSelIdx] comes into effect as follows:

  – If the new value of CpbSize[SchedSelIdx] is greater than the old CPB size, it comes into effect at the initial CPB arrival time of the current access unit.

  – Otherwise, the new value of CpbSize[SchedSelIdx] comes into effect at the CPB removal time of the current access unit.

F.13.2.3 Timing of decoding unit removal and decoding of decoding unit

The variables InitCpbRemovalDelay[SchedSelIdx], InitCpbRemovalDelayOffset[SchedSelIdx], CpbDelayOffset and DpbDelayOffset are derived as follows:

– If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb_delay_offset, DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb_delay_offset and InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_alt_cpb_removal_delay[SchedSelIdx] and nal_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_alt_cpb_removal_delay[SchedSelIdx] and vcl_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause F.13.1:

  – Access unit 0 includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_RADL or BLA_N_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1.

  – Access unit 0 includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_LP or includes a CRA picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 0 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1, and one or more of the following conditions are true:

    – UseAltCpbParamsFlag for access unit 0 is equal to 1.

    – DefaultInitCpbParamsFlag is equal to 0.

  – Access unit 0 includes an IRAP picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 1 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:

    – UseAltCpbParamsFlag for access unit 0 is equal to 1.

    – DefaultInitCpbParamsFlag is equal to 0.

– Otherwise, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause F.13.1, CpbDelayOffset and DpbDelayOffset are both set equal to 0.

The nominal removal time of the access unit n from the CPB is specified as follows:

– If access unit n is the access unit with n equal to 0 (the access unit that initializes the HRD), the nominal removal time of the access unit from the CPB is specified by:

$$\text{AuNominalRemovalTime}[\ 0\ ] = \text{InitCpbRemovalDelay}[\ \text{SchedSelIdx}\ ] \div 90\,000 \quad \text{(F-75)}$$

– Otherwise, the following applies:

  – When access unit n is the first access unit of a buffering period that does not initialize the HRD, the following applies:

  The nominal removal time of the access unit n from the CPB is specified by:

```
if( !concatenationFlag ) {
    baseTime = AuNominalRemovalTime[ firstPicInPrevBuffPeriod ]
    tmpCpbRemovalDelay = AuCpbRemovalDelayVal
```

```
} else {
    baseTime = AuNominalRemovalTime[ prevNonDiscardablePic ]
    tmpCpbRemovalDelay =
        Max( ( auCpbRemovalDelayDeltaMinus1 + 1 ),                                  (F-76)
            Ceil( ( InitCpbRemovalDelay[ SchedSelIdx ] ÷ 90 000 +

    AuFinalArrivalTime[ n − 1 ] − AuNominalRemovalTime[ n − 1 ] ) ÷ ClockTick ) )
}
AuNominalRemovalTime[ n ] = baseTime + ClockTick * ( tmpCpbRemovalDelay −
CpbDelayOffset )
```

where AuNominalRemovalTime[firstPicInPrevBuffPeriod] is the nominal removal time of the first access unit of the previous buffering period, AuNominalRemovalTime[prevNonDiscardablePic] is the nominal removal time of the preceding access unit in decoding order, each picture of which is with TemporalId equal to 0 that is not a RASL, RADL or SLNR picture, AuCpbRemovalDelayVal is the value of AuCpbRemovalDelayVal derived according to au_cpb_removal_delay_minus1 in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit n, and concatenationFlag and auCpbRemovalDelayDeltaMinus1 are the values of the syntax elements concatenation_flag and au_cpb_removal_delay_delta_minus1, respectively, in the buffering period SEI message, selected as specified in clause F.13.1, associated with access unit n.

After the derivation of the nominal CPB removal time and before the derivation of the DPB output time of access unit n, the values of CpbDelayOffset and DpbDelayOffset are updated as follows:

- If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb_delay_offset and DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb_delay_offset, where the buffering period SEI message containing the syntax elements is selected as specified in clause F.13.1:
  - Access unit n includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_RADL or BLA_N_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1.
  - Access unit n includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_LP or includes a CRA picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 0, the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and UseAltCpbParamsFlag for access unit n is equal to 1.
  - Access unit n includes an IRAP picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 1, the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and UseAltCpbParamsFlag for access unit n is equal to 1.
- Otherwise, CpbDelayOffset and DpbDelayOffset are both set equal to 0.

- When access unit n is not the first access unit of a buffering period, the nominal removal time of the access unit n from the CPB is specified by:

```
AuNominalRemovalTime[ n ] = AuNominalRemovalTime[ firstPicInCurrBuffPeriod ] +
        ClockTick * ( AuCpbRemovalDelayVal − CpbDelayOffset )                      (F-77)
```

where AuNominalRemovalTime[firstPicInCurrBuffPeriod] is the nominal removal time of the first access unit of the current buffering period, and AuCpbRemovalDelayVal is the value of AuCpbRemovalDelayVal derived according to au_cpb_removal_delay_minus1 in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit n.

When SubPicHrdFlag is equal to 1, the following applies:

- The variable duCpbRemovalDelayInc is derived as follows:
  - If sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 0, duCpbRemovalDelayInc is set equal to the value of du_spt_cpb_removal_delay_increment in the decoding unit information SEI message, selected as specified in clause F.13.1, associated with decoding unit m.
  - Otherwise, if du_common_cpb_removal_delay_flag is equal to 0, duCpbRemovalDelayInc is set equal to the value of du_cpb_removal_delay_increment_minus1[i] + 1 for decoding unit m in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit n, where the value of i is 0 for the first num_nalus_in_du_minus1[0] + 1 consecutive NAL units in the access unit that contains decoding unit m, 1 for the subsequent num_nalus_in_du_minus1[1] + 1 NAL units in the same access unit, 2 for the subsequent num_nalus_in_du_minus1[2] + 1 NAL units in the same access unit, etc.

– Otherwise, duCpbRemovalDelayInc is set equal to the value of du_common_cpb_removal_delay_increment_minus1 + 1 in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit n.

– The nominal removal time of decoding unit m from the CPB is specified as follows, where AuNominalRemovalTime[n] is the nominal removal time of access unit n:

– If decoding unit m is the last decoding unit in access unit n, the nominal removal time of decoding unit m DuNominalRemovalTime[m] is set equal to AuNominalRemovalTime[n].

– Otherwise (decoding unit m is not the last decoding unit in access unit n), the nominal removal time of decoding unit m DuNominalRemovalTime[m] is derived as follows:

```
if( sub_pic_cpb_params_in_pic_timing_sei_flag )
    DuNominalRemovalTime[ m ] = DuNominalRemovalTime[ m + 1 ] −
        ClockSubTick * duCpbRemovalDelayInc                          (F-78)
else
    DuNominalRemovalTime[ m ] = AuNominalRemovalTime[ n ] −
        ClockSubTick * duCpbRemovalDelayInc
```

If SubPicHrdFlag is equal to 0, the removal time of access unit n from the CPB is specified as follows, where AuFinalArrivalTime[n] and AuNominalRemovalTime[n] are the final CPB arrival time and nominal CPB removal time, respectively, of access unit n:

```
if( !low_delay_hrd_flag[ HighestTid ] | | AuNominalRemovalTime[ n ] >=
AuFinalArrivalTime[ n ] )
    AuCpbRemovalTime[ n ] = AuNominalRemovalTime[ n ]
else                                                                  (F-79)
    AuCpbRemovalTime[ n ] = AuNominalRemovalTime[ n ] + ClockTick *
        Ceil( ( AuFinalArrivalTime[ n ] − AuNominalRemovalTime[ n ] ) ÷ ClockTick )
```

NOTE 1 – When low_delay_hrd_flag[HighestTid] is equal to 1 and AuNominalRemovalTime[n] is less than AuFinalArrivalTime[n], the size of access unit n is so large that it prevents removal at the nominal removal time.

Otherwise (SubPicHrdFlag is equal to 1), the removal time of decoding unit m from the CPB is specified as follows:

– The following applies:

```
if( !low_delay_hrd_flag[ HighestTid ] | | DuNominalRemovalTime[ m ] >=
DuFinalArrivalTime[ m ] )
    DuCpbRemovalTime[ m ] = DuNominalRemovalTime[ m ]                 (F-80)
else
    DuCpbRemovalTime[ m ] = DuFinalArrivalTime[ m ]
```

NOTE 2 – When low_delay_hrd_flag[HighestTid] is equal to 1 and DuNominalRemovalTime[m] is less than DuFinalArrivalTime[m], the size of decoding unit m is so large that it prevents removal at the nominal removal time.

– When cbr_flag[SchedSelIdx] is equal to 0, the following applies:

– Let refDuCpbRemovalTime be equal to the CPB removal time of the previous DU preceding the current DU in decoding order (regardless of the bitstream partitions to which the previous DU and the current DU belong).

– The variable DuCpbRemovalTime[m] is modified as follows:

```
DuCpbRemovalTime[ m ] = Max( DuCpbRemovalTime[ m ], refDuCpbRemovalTime )
    (F-81)
```

If SubPicHrdFlag is equal to 0, at the CPB removal time of access unit n, the access unit is instantaneously decoded.

Otherwise (SubPicHrdFlag is equal to 1), at the CPB removal time of decoding unit m, the decoding unit is instantaneously decoded, and when decoding unit m is the last decoding unit of access unit n, the following applies:

– Access unit n is considered as decoded.

– The final CPB arrival time of access unit n, i.e., AuFinalArrivalTime[n], is set equal to the final CPB arrival time of the last decoding unit in access unit n, i.e., DuFinalArrivalTime[m].

– The nominal CPB removal time of access unit n, i.e., AuNominalRemovalTime[n], is set equal to the nominal CPB removal time of the last decoding unit in access unit n, i.e., DuNominalRemovalTime[m].

– The CPB removal time of access unit n, i.e., AuCpbRemovalTime[m], is set equal to the CPB removal time of the last decoding unit in access unit n, i.e., DuCpbRemovalTime[m].

F.13.3 Operation of decoded picture buffer

F.13.3.1 General

The specifications in this clause apply independently to each set of decoded picture buffer (DPB) parameters selected as specified in clause F.13.1.

The decoded picture buffer consists of sub-DPBs and each sub-DPB contains picture storage buffers for storage of decoded pictures of one layer. Each of the picture storage buffers of a sub-DPB may contain a decoded picture that is marked as "used for reference" or is held for future output.

The processes specified in clauses F.13.3.2, F.13.3.3, F.13.3.4 and F.13.3.5 are sequentially applied as specified below, and are applied independently for each layer, starting from the base layer, in increasing order of nuh_layer_id values of the layers in the bitstream. When these processes are applied for a particular layer, only the sub-DPB for the particular layer is affected. In the descriptions of these processes, the DPB refers to the sub-DPB for the particular layer, and the particular layer is referred to as the current layer.

NOTE – In the operation of output timing DPB, decoded pictures with PicOutputFlag equal to 1 in the same access unit are output consecutively in ascending order of the nuh_layer_id values of the decoded pictures.

Let picture n and the current picture be the coded picture or decoded picture of the access unit n for a particular value of nuh_layer_id, wherein n is a non-negative integer number.

F.13.3.2 Removal of pictures from the DPB before decoding of the current picture

When the current picture is not picture 0 in the current layer, the removal of pictures in the current layer, with nuh_layer_id equal to currLayerId, from the DPB before decoding of the current picture, i.e., picture n, but after parsing the slice header of the first slice of the current picture, happens instantaneously at the CPB removal time of the first decoding unit of the current picture and proceeds as follows:

– The decoding process for RPS as specified in clause F.8.3.2 is invoked.

– The variable listOfSubDpbsToEmpty is derived as follows:

  – If a new VPS is activated by the current access unit or the current picture is an IRAP picture with nuh_layer_id equal to SmallestLayerId, NoRaslOutputFlag equal to 1 and NoClrasOutputFlag equal to 1, listOfSubDpbsToEmpty is set to include all the sub-DPBs.

  – Otherwise, if the current picture is an IRAP picture with any nuh_layer_id value indepLayerId such that NumDirectRefLayers[indepLayerId] is equal to 0 and indepLayerId is greater than SmallestLayerId, and with NoRaslOutputFlag equal to 1, and LayerResetFlag is equal to 1, listOfSubDpbsToEmpty is set equal to the sub-DPBs containing the current layer and the sub-DPBs containing the predicted layers of the current layer.

  – Otherwise, listOfSubDpbsToEmpty is set equal to the sub-DPB containing the current layer.

– When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and any of the following conditions is true:

  – nuh_layer_id equal to SmallestLayerId,

  – nuh_layer_id of the current layer is greater than SmallestLayerId and NumDirectRefLayers[nuh_layer_id] is equal to 0,

  the following ordered steps are applied:

  1. The variable NoOutputOfPriorPicsFlag is derived for the decoder under test as follows:

     – If the current picture is a CRA picture, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no_output_of_prior_pics_flag).

     – Otherwise, if the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, separate_colour_plane_flag or sps_max_dec_pic_buffering_minus1[HighestTid] derived from the active SPS for the current layer is different from the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, separate_colour_plane_flag or sps_max_dec_pic_buffering_minus1[HighestTid], respectively, derived from the SPS that was active for the current layer when decoding the preceding picture in the current layer, NoOutputOfPriorPicsFlag may (but should not) be set equal to 1 by the decoder under test, regardless of the value of no_output_of_prior_pics_flag.

NOTE – Although setting NoOutputOfPriorPicsFlag equal to no_output_of_prior_pics_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.

– Otherwise, NoOutputOfPriorPicsFlag is set equal to no_output_of_prior_pics_flag.

2. When the value of NoOutputOfPriorPicsFlag derived for the decoder under test is equal to 1, all non-empty picture storage buffers in all the sub-DPBs contained in listOfSubDpbsToEmpty are emptied without output of the pictures they contain, and the sub-DPB fullness of each sub-DPB in listOfSubDpbsToEmpty is set equal to 0.

– When both of the following conditions are true for any pictures k in the DPB, all such pictures k in the DPB are removed from the DPB:

  – picture k is marked as "unused for reference".

  – picture k has PicOutputFlag equal to 0 or its DPB output time is less than or equal to the CPB removal time of the first decoding unit (denoted as decoding unit m) of the current picture n; i.e., DpbOutputTime[k] is less than or equal to DuCpbRemovalTime[m].

– For each picture that is removed from the DPB, the DPB fullness is decremented by one.

F.13.3.3 Picture output

The processes specified in this clause happen instantaneously at the CPB removal time of access unit n, AuCpbRemovalTime[n].

When access unit n has AuOutputFlag equal to 1, its DPB output time DpbOutputTime[n] is derived as follows, where the variable firstPicInBufferingPeriodFlag is equal to 1 if access unit n is the first access unit of a buffering period and 0 otherwise:

```
if( !SubPicHrdFlag ) {
    DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockTick * picDpbOutputDelay        (F-82)
    if( firstPicInBufferingPeriodFlag )
        DpbOutputTime[ n ] −= ClockTick * DpbDelayOffset
} else
    DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockSubTick * picSptDpbOutputDuDelay
```

where picDpbOutputDelay is the value of pic_dpb_output_delay in the picture timing SEI message associated with access unit n, and picSptDpbOutputDuDelay is the value of pic_spt_dpb_output_du_delay, when present, in the decoding unit information SEI messages associated with access unit n, or the value of pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n when there is no decoding unit information SEI message associated with access unit n or no decoding unit information SEI message associated with access unit n has pic_spt_dpb_output_du_delay present.

NOTE – When the syntax element pic_spt_dpb_output_du_delay is not present in any decoding unit information SEI message associated with access unit n, the value is inferred to be equal to pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n.

The output of the current picture contained in access unit n is specified as follows:

– If PicOutputFlag is equal to 1 and DpbOutputTime[n] is equal to AuCpbRemovalTime[n], the current picture is output.

– Otherwise, if PicOutputFlag is equal to 0, the current picture is not output, but will be stored in the DPB as specified in clause F.13.3.4.

– Otherwise (PicOutputFlag is equal to 1 and DpbOutputTime[n] is greater than AuCpbRemovalTime[n]), the current picture is output later and will be stored in the DPB (as specified in clause F.13.3.4) and is output at time DpbOutputTime[n] unless indicated not to be output by NoOutputOfPriorPicsFlag equal to 1.

When output, a picture is cropped, using the conformance cropping window specified in the active SPS for the layer containing the picture.

When access unit n is an access unit that has AuOutputFlag equal to 1 and is not the last access unit of the bitstream that has AuOutputFlag equal to 1, the value of the variable DpbOutputInterval[n] is derived as follows:

$$\text{DpbOutputInterval[n]} = \text{DpbOutputTime[nextAuInOutputOrder]} - \text{DpbOutputTime[n]} \qquad \text{(F-83)}$$

where nextAuInOutputOrder is the access unit that follows access unit n in output order and has AuOutputFlag equal to 1.

F.13.3.4 Current decoded picture marking and storage

The current decoded picture after the invocation of the in-loop filter process as specified in clause F.8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one and this picture is marked as "used for short-term reference".

When TwoVersionsOfCurrDecPicFlag is equal to 1, the current decoded picture before the invocation of the in-loop filter process as specified in clause F.8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one, and this picture is marked as "used for long-term reference".

NOTE – Unless more memory than required by the level limit is available for storage of decoded pictures, decoders should start storing decoded parts of the current picture into the DPB when the first slice segment is decoded and continue storing more decoded samples as the decoding process proceeds.

F.13.3.5 Removal of pictures from the DPB after decoding of the current picture

When TwoVersionsOfCurrDecPicFlag is equal to 1, immediately after decoding of the current picture, at the CPB removal time of the last decoding unit of access unit n (containing the current picture), the current decoded picture that is marked as "used for long-term reference" is removed from the DPB, and the DPB fullness is decremented by one.

F.13.4 Bitstream conformance

A bitstream of coded data conforming to this Specification shall fulfil all requirements specified in this clause.

The bitstream shall be constructed according to the syntax, semantics and constraints specified in this Specification outside of this annex.

The first access unit in a bitstream shall be an IRAP access unit.

When vps_base_layer_internal_flag is equal to 0, all the following bitstream conformance constraints apply only to coded pictures present in the bitstream and do not apply to pictures with nuh_layer_id equal to 0 which are provided by external means.

Let currPicLayerId be equal to the nuh_layer_id of the current picture.

For each current picture, let the variables maxPicOrderCnt and minPicOrderCnt be set equal to the maximum and the minimum, respectively, of the PicOrderCntVal values of the following pictures with nuh_layer_id equal to currPicLayerId:

– The current picture.

– The previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL or SLNR picture.

– The short-term reference pictures in the RPS of the current picture.

– All pictures m that have PicOutputFlag equal to 1, AuCpbRemovalTime[m] less than AuCpbRemovalTime[currAu] and DpbOutputTime[m] greater than or equal to AuCpbRemovalTime[currAu], where currAu is the access unit containing the current picture, and AuCpbRemovalTime[m] and DpbOutputTime[m] are the CPB removal time and the DPB output time, respectively, of the access unit containing picture m.

The bitstream and the sub-bitstream of each output layer set are tested by the HRD for conformance as specified in clause C.1 and clause F.13.1. All the conditions specified in clause C.4 shall be fulfilled for each of the first and second sets of conformance tests. All of the following conditions shall be fulfilled for each of the third set of conformance tests:

1. For each partition unit n, with n greater than 0, associated with a buffering period SEI message, let the variable deltaTime90k[n] be specified as follows:

$$\text{deltaTime90k[n]} = 90\,000 * (\text{ AuNominalRemovalTime[n]} - \text{AuFinalArrivalTime[n} - 1\text{] }) \quad \text{(F-84)}$$

   The value of InitCpbRemovalDelay[SchedSelIdx] is constrained as follows:

   – If cbr_flag[SchedSelIdx] is equal to 0, the following condition shall be true:

$$\text{InitCpbRemovalDelay[SchedSelIdx]} <= \text{Ceil(deltaTime90k[n])} \quad \text{(F-85)}$$

   – Otherwise (cbr_flag[SchedSelIdx] is equal to 1), the following condition shall be true:

$$\text{Floor(deltaTime90k[n])} <= \text{InitCpbRemovalDelay[SchedSelIdx]} <= \text{Ceil(deltaTime90k[n])} \quad \text{(F-86)}$$

   NOTE 1 – The exact number of bits in the BPB at the removal time of each partition unit may depend on which buffering period SEI message is selected to initialize the HRD. Encoders must take this into account to ensure that all

specified constraints must be obeyed regardless of which buffering period SEI message is selected to initialize the HRD, as the HRD may be initialized at any one of the buffering period SEI messages.

2. A BPB overflow is specified as the condition in which the total number of bits in the BPB is greater than the BPB size. The BPB shall never overflow.

3. When low_delay_hrd_flag[HighestTid] is equal to 0, the BPB shall never underflow. A BPB underflow is specified as follows:
   – If SubHrdFlag is equal to 0, a BPB underflow is specified as the condition in which the nominal BPB removal time of partition unit n AuNominalRemovalTime[n] is less than the final BPB arrival time of partition unit n AuFinalArrivalTime[N] for at least one value of n.
   – Otherwise (SubHrdFlag is equal to 1), a BPB underflow is specified as the condition in which the nominal BPB removal time of decoding unit m DuNominalRemovalTime[m] is less than the final BPB arrival time of decoding unit m DuFinalArrivalTime[m] for at least one value of m.

4. When SubPicHrdFlag is equal to 1, low_delay_hrd_flag[HighestTid] is equal to 1 and the nominal removal time of a decoding unit m of partition unit n is less than the final BPB arrival time of decoding unit m (i.e., DuNominalRemovalTime[m] < DuFinalArrivalTime[m]), the nominal removal time of partition unit n shall be less than the final BPB arrival time of partition unit n (i.e., AuNominalRemovalTime[n] < AuFinalArrivalTime[n]).

5. The nominal removal times of partition units from the BPB (starting from the second partition unit in decoding order) shall satisfy the constraints on AuNominalRemovalTime[n] and AuCpbRemovalTime[n] expressed in clauses A.4.1 through A.4.2 for bitstreams or layers conforming to profiles defined in Annex A, or expressed in clauses G.11.2.1 through G.11.2.2 for bitstreams or layers conforming to profiles defined in Annex G, or expressed in clauses H.11.2.1 through H.11.2.2 for bitstreams or layers conforming to profiles defined in Annex H.

6. For each current picture, after invocation of the process for removal of pictures from the sub-DPB as specified in clause F.13.3.2, the number of decoded pictures in the sub-DPB for the current layer, including all pictures n, where each picture n is contained in partition unit k_n, in the current layer that are marked as "used for reference", or that have PicOutputFlag equal to 1 and AuCpbRemovalTime[k_n] less than AuCpbRemovalTime[currAu], where currAu is the partition unit containing the current picture, shall be less than or equal to max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid], where layerIdx is equal to the value such that LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx] is equal to currPicLayerId.

7. All reference pictures shall be present in the DPB when needed for prediction. Each picture that has PicOutputFlag equal to 1 shall be present in the DPB at its DPB output time unless it is removed from the DPB before its output time by one of the processes specified in clause F.13.3.

8. For each current picture that is not an IRAP picture with NoRaslOutputFlag equal to 1 or that is not the first picture of the current layer with nuh_layer_id greater than 0 that follows an IRAP picture that has nuh_layer_id equal to 0 and has NoClrasOutputFlag equal to 1, the value of maxPicOrderCnt − minPicOrderCnt shall be less than MaxPicOrderCntLsb / 2.

9. The value of DpbOutputInterval[n] as given by Equation F-83, which is the difference between the output time of a partition unit that has AuOutputFlag equal to 1 and that of the first partition unit following it in output order and having AuOutputFlag equal to 1, shall satisfy the constraint on DpbOutputInterval[n] expressed in clause A.4.2 for the profile, tier and level of the bitstream or layer using the decoding process specified in clauses 2 through 10, or expressed in clause G.11.2.2 for the profile, tier and level of the bitstream or layer using the decoding process specified in clauses F.2 through F.10 and either clauses G.2 through G.10 or clauses H.2 through H.10.

10. For each current picture, when sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 1, let tmpCpbRemovalDelaySum be derived as follows:

```
tmpCpbRemovalDelaySum = 0
for( i = 0; i < num_decoding_units_minus1; i++ )                                  (F-87)
    tmpCpbRemovalDelaySum  +=  du_cpb_removal_delay_increment_minus1[ i ] + 1
```

    The value of ClockSubTick * tmpCpbRemovalDelaySum shall be equal to the difference between the nominal BPB removal time of the current partition unit and the nominal BPB removal time of the first decoding unit in the current partition unit in decoding order.

11. Let picA be an IRAP picture with NoRaslOutputFlag equal to 1 and belonging to a layer layerA. Let auB be the earlier, in decoding order, of the first partition unit containing an IRAP picture with nuh_layer_id equal to 0 and NoClrasOutputFlag equal to 1 that succeeds picA in decoding order and the first partition unit containing an IRAP picture with NoRaslOutputFlag equal to 1 in layerA that succeeds picA in decoding order. For any two pictures picM and picN in the layer layerA contained in partition units m and n, respectively, that either are picA or succeed

picA in decoding order and precede auB in decoding order, when DpbOutputTime[m] is greater than DpbOutputTime[n], PicOrderCnt(picM) shall be greater than PicOrderCnt(picN), where PicOrderCnt(picM) and PicOrderCnt(picN) are the PicOrderCntVal values of picM and picN, respectively, immediately after the invocation of the decoding process for picture order count of the latter of picM and picN in decoding order.

NOTE 2 – All pictures of an earlier CVS in decoding order that are output are output before any pictures of a later CVS in decoding order. Within any particular CVS where all pictures have poc_reset_id not present or equal to 0, the pictures that are output are output in increasing PicOrderCntVal order. For any particular CVS where some pictures have poc_reset_id greater than 0, within each POC resetting period, the pictures that are output are output in increasing PicOrderCntVal order, and for any two pictures that are output and are in different POC resetting periods, the one that is earlier in decoding order is output earlier.

12. For any decoding unit m, DuCpbRemovalTime[m] shall be greater than or equal to the BPB removal time of the previous DU that precedes the current DU in decoding order and that is in the same bitstream partition as the decoding unit m or includes one or more VCL NAL units of a picture that may be used as a direct or indirect reference picture for the picture containing the one or more VCL NAL units included in the decoding unit m.

13. The DPB output times derived for all pictures in any particular access unit shall be the same.

F.13.5 Decoder conformance

F.13.5.1 General

A decoder conforming to this Specification shall fulfil all requirements specified in this clause.

A decoder claiming conformance to a specific profile, tier and level shall be able to successfully decode all bitstreams that conform to the bitstream conformance requirements specified in clause F.13.4, in the manner specified in Annexes A, G and H for profile, tier and level specified in Annexes A, G and H, respectively, provided that all VPSs, SPSs and PPSs referred to by the VCL NAL units, appropriate buffering period, picture timing and decoding unit information SEI messages are conveyed to the decoder, in a timely manner, either in the bitstream (by non-VCL NAL units), or by external means not specified in this Specification, and, when vps_base_layer_internal_flag is equal to 0, the decoded pictures with nuh_layer_id equal to 0 and their properties as specified in clause F.8.1 and its subclauses are conveyed to the decoder in a timely manner by external means not specified in this Specification.

When a bitstream contains syntax elements that have values that are specified as reserved and it is specified that decoders shall ignore values of the syntax elements or NAL units containing the syntax elements having the reserved values, and the bitstream is otherwise conforming to this Specification, a conforming decoder shall decode the bitstream in the same manner as it would decode a conforming bitstream and shall ignore the syntax elements or the NAL units containing the syntax elements having the reserved values as specified.

There are two types of conformance that can be claimed by a decoder: output timing conformance and output order conformance.

To check conformance of a decoder, test bitstreams containing one or more bitstream partitions conforming to the claimed profile, tier and level, as specified in clause F.13.4 are delivered by a hypothetical stream scheduler (HSS) both to the HRD and to the decoder under test (DUT). When vps_base_layer_internal_flag is equal to 0, decoded pictures with nuh_layer_id equal to 0 and their properties as specified in clause F.8.1 and its subclauses are also conveyed both to the HRD and to the DUT in a timely manner by external means not specified in this Specification. All cropped decoded pictures output by the HRD shall also be output by the DUT, each cropped decoded picture output by the DUT shall be a picture with PicOutputFlag equal to 1, and, for each such cropped decoded picture output by the DUT, the values of all samples that are output shall be equal to the values of the samples produced by the specified decoding process. The flag BaseLayerOutputFlag and all flags BaseLayerPicOutputFlag output by the HRD shall also be output by the DUT, and the values that are output shall be equal to the values produced by the specified decoding process.

For output timing decoder conformance, the HSS operates as described above, with delivery schedules selected only from the subset of values of SchedSelIdx for which the bit rate and CPB size are restricted as specified in Annexes A, G and H for the specified profile, tier and level in Annexes A, G and H, respectively, or with "interpolated" delivery schedules as specified below for which the bit rate and CPB size are restricted as specified in Annexes A, G and H for the specified profile, tier and level in Annexes A, G and H, respectively. The same delivery schedule is used for both the HRD and the DUT.

When the HRD parameters and the buffering period SEI messages are present with the number of signalled delivery schedules greater than 0, the decoder shall be capable of decoding each bitstream partition as delivered from the HSS operating using an "interpolated" delivery schedule specified as having peak bit rate r, CPB size c(r) and initial CPB removal delay (f(r) ÷ r) as follows:

$$\alpha = (r - \text{BitRate}[\text{SchedSelIdx} - 1]) \div (\text{BitRate}[\text{SchedSelIdx}] - \text{BitRate}[\text{SchedSelIdx} - 1]), \quad \text{(F-88)}$$

$$c(r) = \alpha * CpbSize[SchedSelIdx] + (1 - \alpha) * CpbSize[SchedSelIdx - 1], \quad \text{(F-89)}$$

$$f(r) = \alpha * InitCpbRemovalDelay[SchedSelIdx] * BitRate[SchedSelIdx] + (1 - \alpha) * InitCpbRemovalDelay[SchedSelIdx - 1] * BitRate[SchedSelIdx - 1] \quad \text{(F-90)}$$

for any SchedSelIdx > 0 and r such that BitRate[SchedSelIdx − 1] <= r <= BitRate[SchedSelIdx] such that r and c(r) are within the limits as specified in Annexes A, G and H for the maximum bit rate and buffer size for the specified profile, tier and level defined in Annexes A, G and H, respectively. The following applies for the specifications above:

– Let combIdx be any value in the range of 0 to num_bsp_schedules_minus1[TargetOlsIdx][TargetPsIdx][HighestTid], inclusive.

– SchedSelIdx shall be one of the values among bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][combIdx][j] for and j is the index of the bitstream partition index of TargetBitstreamPartition.

– SchedSelIdx − 1 is to be understood as bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][combIdx − 1][j] for any value of k in the range of 1 to combIdx, inclusive.

NOTE 1 – InitCpbRemovalDelay[SchedSelIdx] can be different from one buffering period to another and need to be recalculated.

For output timing decoder conformance, an HRD as described above is used and the timing (relative to the delivery time of the first bit) of picture output is the same for both the HRD and the DUT up to a fixed delay.

For output order decoder conformance, the following applies for each bitstream partition in TargetPartitioningScheme:

– The HSS delivers the TargetBitstreamPartition to the DUT "by demand" from the DUT, meaning that the HSS delivers bits (in decoding order) only when the DUT requires more bits to proceed with its processing.

  NOTE 2 – This means that for this test, the coded picture buffer of the DUT could be as small as the size of the largest decoding unit.

– A modified HRD as described below is used, and the HSS delivers the bitstream to the HRD by one of the schedules specified in the TargetBitstreamPartition such that the bit rate and CPB size are restricted as specified in Annex A for profiles defined in Annex A, Annex G for profiles defined in Annex G and Annex H for profiles defined in Annex H. The order of pictures output shall be the same for both the HRD and the DUT.

– The HRD BPB size is given by CpbSize[SchedSelIdx] as specified in clause F.13.1. The sub-DPB size of the sub-DPB for a layer with nuh_layer_id equal to currLayerId is max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid] + 1, where layerIdx is equal to the value such that LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx] is equal to currLayerId. The removal time from the CPB for the HRD is the final bit arrival time and decoding is immediate. The operation of the DPB of this HRD is as described in the subclauses of clause F.13.5.2.

F.13.5.2 Operation of the output order DPB

F.13.5.2.1 General

The decoded picture buffer consists of sub-DPBs and each sub-DPB contains picture storage buffers for storage of decoded pictures of one layer. Each of the picture storage buffers of a sub-DPB contains a decoded picture that is marked as "used for reference" or is held for future output.

The process for output and removal of pictures from the DPB before decoding of the current picture as specified in clause F.13.5.2.2 is invoked, followed by the invocation of the process for current decoded picture marking and storage as specified in clause F.13.3.4, further followed by the invocation of the process for removal of pictures from the DPB after decoding of the current picture as specified in clause F.13.3.5, and finally followed by the invocation of the process for additional bumping as specified in clause F.13.5.2.3. The "bumping" process is specified in clause F.13.5.2.4 and is invoked as specified in clauses F.13.5.2.2 and F.13.5.2.3.

These processes are applied independently for each layer, starting from the base layer, in increasing order of the nuh_layer_id values of the layers in the bitstream. When these processes are applied for a particular layer, only the sub-DPB for the particular layer is affected except for the "bumping" process, which may crop and output pictures, mark pictures as "not needed for output" and empty picture storage buffers for any layer.

NOTE – In the operation of output order DPB, same as in the operation of output timing DPB, decoded pictures with PicOutputFlag equal to 1 in the same access unit are also output consecutively in ascending order of the nuh_layer_id values of the decoded pictures.

Let picture n and the current picture be the coded picture or decoded picture of the access unit n for a particular value of nuh_layer_id, wherein n is a non-negative integer number.

When these processes are applied for a layer with nuh_layer_id equal to currLayerId, the variables MaxNumReorderPics, MaxLatencyIncreasePlus1, MaxLatencyValue and MaxDecPicBufferingMinus1 are derived as follows:

– MaxNumReorderPics is set equal to max_vps_num_reorder_pics[TargetOlsIdx][HighestTid] of the active VPS.

– MaxLatencyIncreasePlus1 is set equal to the value of the syntax element max_vps_latency_increase_plus1[TargetOlsIdx][HighestTid] of the active VPS.

– MaxLatencyValue is set equal to MaxVpsLatencyPictures[TargetOlsIdx][HighestTid] of the active VPS.

– MaxDecPicBufferingMinus1 is set equal to the value of the syntax element max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid] of the active VPS, where layerIdx is equal to the value such that LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx] is equal to currLayerId.

F.13.5.2.2 Output and removal of pictures from the DPB before decoding of the current picture

When the current picture is not picture 0 in the current layer, the output and removal of pictures in the current layer, with nuh_layer_id equal to currLayerId, from the DPB before the decoding of the current picture, i.e., picture n, but after parsing the slice header of the first slice of the current picture and before the invocation of the decoding process for picture order count, happens instantaneously when the first decoding unit of the current picture is removed from the CPB and proceeds as follows:

– When the current picture is a POC resetting picture, all pictures in the DPB that do not belong to the current access unit and that are marked as "needed for output" are output, starting with pictures with the smallest value of PicOrderCntVal of all pictures excluding those in the current access unit in the DPB, in ascending order of the PicOrderCntVal values, and pictures with the same value of PicOrderCntVal are output in ascending order of the nuh_layer_id values. When a picture is output, it is cropped using the conformance cropping window specified in the active SPS for the picture, the cropped picture is output, and the picture is marked as "not needed for output".

– The decoding processes for picture order count and RPS are invoked. When decoding a CVS conforming to one or more of the profiles specified in Annex A using the decoding process specified in clauses 2 through 10, the decoding processes for picture order count and RPS that are invoked are as specified in clauses 8.3.1 and 8.3.2, respectively. When decoding a CVS conforming to one or more of the profiles specified in Annex G or H using the decoding process specified in Annex F, and Annex G or H, the decoding processes for picture order count and RPS that are invoked are as specified in clauses F.8.3.1 and F.8.3.2, respectively.

– The variable listOfSubDpbsToEmpty is derived as follows:

  – If a new VPS is activated by the current access unit or the current picture is IRAP picture with nuh_layer_id equal to SmallestLayerId, NoRaslOutputFlag equal to 1 and NoClrasOutputFlag equal to 1, listOfSubDpbsToEmpty is set equal to all the sub-DPBs.

  – Otherwise, if the current picture is an IRAP picture with any nuh_layer_id value indepLayerId such that NumDirectRefLayers[indepLayerId] is equal to 0 and indepLayerId is greater than SmallestLayerId, and with NoRaslOutputFlag equal to 1, and LayerResetFlag is equal to 1, listOfSubDpbsToEmpty is set equal to the sub-DPBs containing the current layer and the sub-DPBs containing the predicted layers of the current layer.

  – Otherwise, listOfSubDpbsToEmpty is set equal to the sub-DPB containing the current layer.

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and any of the following conditions is true:

  – nuh_layer_id equal to SmallestLayerId,

  – nuh_layer_id of the current layer is greater than SmallestLayerId and NumDirectRefLayers[nuh_layer_id] is equal to 0,

  the following ordered steps are applied:

  1. The variable NoOutputOfPriorPicsFlag is derived for the decoder under test as follows:

     – If the current picture is a CRA picture, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no_output_of_prior_pics_flag).

     – Otherwise, if the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, separate_colour_plane_flag or sps_max_dec_pic_buffering_minus1[HighestTid] derived from the active SPS for the current layer is different from the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, bit_depth_luma_minus8, bit_depth_chroma_minus8, separate_colour_plane_flag or sps_max_dec_pic_buffering_minus1[HighestTid], respectively, derived from the SPS that was active for the current layer when decoding the preceding picture in the current layer, NoOutputOfPriorPicsFlag may (but should not) be set equal to 1 by the decoder under test, regardless of the value of no_output_of_prior_pics_flag.

       NOTE – Although setting NoOutputOfPriorPicsFlag equal to no_output_of_prior_pics_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.

– Otherwise, NoOutputOfPriorPicsFlag is set equal to no_output_of_prior_pics_flag.

2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD as follows:

   – If NoOutputOfPriorPicsFlag is equal to 0, all non-empty picture storage buffers in all the sub-DPBs included in listOfSubDpbsToEmpty are output by repeatedly invoking the "bumping" process specified in clause F.13.5.2.4 until all these pictures are marked as "not needed for output".

   – Otherwise (NoOutputOfPriorPicsFlag is equal to 1), all picture storage buffers containing a picture that is marked as "not needed for output" and "unused for reference" are emptied (without output), all pictures that are contained in a sub-DPB included in listOfSubDpbsToEmpty are emptied, and the sub-DPB fullness of each sub-DPB is decremented by the number of picture storage buffers emptied in that sub-DPB.

– Otherwise, all picture storage buffers that contain a picture in the current layer and that are marked as "not needed for output" and "unused for reference" are emptied (without output). For each picture storage buffer that is emptied, the sub-DPB fullness is decremented by one. When one or more of the following conditions are true, the "bumping" process specified in clause F.13.5.2.4 is invoked repeatedly until none of the following conditions are true:

  – The number of access units that contain at least one decoded picture in the DPB marked as "needed for output" is greater than MaxNumReorderPics.

  – MaxLatencyIncreasePlus1 is not equal to 0 and there is at least one access unit that contains at least one decoded picture in the DPB marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to MaxLatencyValue.

  – The number of pictures in the sub-DPB is greater than or equal to MaxDecPicBufferingMinus1 + 1 − TwoVersionsOfCurrDecPicFlag.

F.13.5.2.3 Additional bumping

The processes specified in this clause happen instantaneously when the last decoding unit of picture n is removed from the CPB.

When the current picture is the last picture in an access unit, the following applies for each decoded picture with nuh_layer_id greater than or equal to (vps_base_layer_internal_flag ? 0 : 1) of the access unit:

– If the decoded picture has PicOutputFlag equal to 1, it is marked as "needed for output".

– Otherwise (the decoded picture has PicOutputFlag equal to 0), it is marked as "not needed for output".

NOTE – Prior to investigating the conditions above, PicOutputFlag of each picture of the access unit is updated as specified in clause F.8.1.2.

When one or more of the following conditions are true, the "bumping" process specified in clause F.13.5.2.4 is invoked repeatedly until none of the following conditions are true:

– The number of access units that contain at least one decoded picture in the DPB marked as "needed for output" is greater than MaxNumReorderPics.

– MaxLatencyIncreasePlus1 is not equal to 0 and there is at least one access unit that contains at least one decoded picture in the DPB marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to MaxLatencyValue.

F.13.5.2.4 "Bumping" process

The "bumping" process consists of the following ordered steps:

1. The picture or pictures that are first for output are selected as the ones having the smallest value of PicOrderCntVal of all pictures in the DPB marked as "needed for output".
2. Each of these pictures is, in ascending nuh_layer_id order, cropped, using the conformance cropping window specified in the active SPS for the picture, the cropped picture is output and the picture is marked as "not needed for output".
3. Each picture storage buffer that contains a picture marked as "unused for reference" and that was one of the pictures cropped and output is emptied and the fullness of the associated sub-DPB is decremented by one.

F.13.6 Demultiplexing process for deriving a bitstream partition

Inputs to this process are a bitstream, a layer identifier list bspLayerId[bspIdx] and the number of layer identifiers numBspLayers in the layer index list bspLayerId[bspIdx].

Output of this process is a bitstream partition.

Let the variable minBspLayerId be the smallest value of bspLayerId[bspIdx] with any value of bspIdx in the range of 0 to numBspLayers − 1, inclusive.

The output bitstream partition consists of selected NAL units of the input bitstream in the same order as they appear in the input bitstream. The following NAL units of the input bitstream are omitted from the output bitstream partition, while the remaining NAL units of the input bitstream are included in the output bitstream partition:

– Omit all NAL units that have a nuh_layer_id value other than bspLayerId[bspIdx] with any value of bspIdx in the range of 0 to numBspLayers − 1, inclusive.

– Omit all SEI NAL units containing a scalable nesting SEI message for which no derived nestingLayerIdList[i] contains any layer identifier value equal to bspLayerId[bspIdx] with any value of bspIdx in the range of 0 to numBspLayers − 1, inclusive.

– Omit all SEI NAL units containing a scalable nesting SEI message for which a derived nestingLayerIdList[i] contains a layer identifier value less than minBspLayerId.

F.14 Supplemental enhancement information

F.14.1 General

The specifications in clause D.1 apply.

F.14.2 SEI payload syntax

F.14.2.1 General SEI payload syntax

The specifications in clause D.2.1 apply.

F.14.2.2 Annex D SEI message syntax for multi-layer extensions

The specifications in clauses D.2.2 through D.2.49 apply.

F.14.2.3 Layers not present SEI message syntax

layers_not_present(payloadSize) {	**Descriptor**
lnp_sei_active_vps_id	u(4)
for(i = 0; i <= MaxLayersMinus1; i++)	
layer_not_present_flag[i]	u(1)
}	

F.14.2.4 Inter-layer constrained tile sets SEI message syntax

inter_layer_constrained_tile_sets(payloadSize) {	**Descriptor**
il_all_tiles_exact_sample_value_match_flag	u(1)
il_one_tile_per_tile_set_flag	u(1)
if(!il_one_tile_per_tile_set_flag) {	
il_num_sets_in_message_minus1	ue(v)
if(il_num_sets_in_message_minus1)	
skipped_tile_set_present_flag	u(1)
numSignificantSets = il_num_sets_in_message_minus1 − skipped_tile_set_present_flag + 1	
for(i = 0; i < numSignificantSets; i++) {	
ilcts_id[i]	ue(v)
il_num_tile_rects_in_set_minus1[i]	ue(v)
for(j = 0; j <= il_num_tile_rects_in_set_minus1[i]; j++) {	
il_top_left_tile_idx[i][j]	ue(v)
il_bottom_right_tile_idx[i][j]	ue(v)
}	
ilc_idc[i]	u(2)
if(!il_all_tiles_exact_sample_value_match_flag)	
il_exact_sample_value_match_flag[i]	u(1)
}	
} else	
all_tiles_ilc_idc	u(2)
}	

F.14.2.5 Bitstream partition nesting SEI message syntax

bsp_nesting(payloadSize) {	**Descriptor**
sei_ols_idx	ue(v)
sei_partitioning_scheme_idx	ue(v)
bsp_idx	ue(v)
num_seis_in_bsp_minus1	ue(v)
while(!byte_aligned( ))	
bsp_nesting_zero_bit /* equal to 0 */	u(1)
for(i = 0; i <= num_seis_in_bsp_minus1; i++)	
sei_message( )	
}	

F.14.2.6 Bitstream partition initial arrival time SEI message syntax

bsp_initial_arrival_time(payloadSize) {	**Descriptor**
psIdx = sei_partitioning_scheme_idx	
if(nalInitialArrivalDelayPresent)	
for(i = 0; i < BspSchedCnt[sei_ols_idx][psIdx][MaxTemporalId[0]]; i++)	
nal_initial_arrival_delay[i]	u(v)
if(vclInitialArrivalDelayPresent)	
for(i = 0; i < BspSchedCnt[sei_ols_idx][psIdx][MaxTemporalId[0]]; i++)	
vcl_initial_arrival_delay[i]	u(v)
}	

F.14.2.7 Sub-bitstream property SEI message syntax

sub_bitstream_property(payloadSize) {	**Descriptor**
sb_property_active_vps_id	u(4)
num_additional_sub_streams_minus1	ue(v)
for(i = 0; i <= num_additional_sub_streams_minus1; i++) {	
sub_bitstream_mode[i]	u(2)
ols_idx_to_vps[i]	ue(v)
highest_sublayer_id[i]	u(3)
avg_sb_property_bit_rate[i]	u(16)
max_sb_property_bit_rate[i]	u(16)
}	
}	

F.14.2.8 Alpha channel information SEI message syntax

alpha_channel_info(payloadSize) {	**Descriptor**
alpha_channel_cancel_flag	u(1)
if(!alpha_channel_cancel_flag) {	
alpha_channel_use_idc	u(3)
alpha_channel_bit_depth_minus8	u(3)
alpha_transparent_value	u(v)
alpha_opaque_value	u(v)
alpha_channel_incr_flag	u(1)
alpha_channel_clip_flag	u(1)
if(alpha_channel_clip_flag)	
alpha_channel_clip_type_flag	u(1)
}	
}	

F.14.2.9 Overlay information SEI message syntax

overlay_info(payloadSize) {	**Descriptor**
overlay_info_cancel_flag	u(1)
if(!overlay_info_cancel_flag) {	
overlay_content_aux_id_minus128	ue(v)
overlay_label_aux_id_minus128	ue(v)
overlay_alpha_aux_id_minus128	ue(v)
overlay_element_label_value_length_minus8	ue(v)
num_overlays_minus1	ue(v)
for(i = 0; i <= num_overlays_minus1; i++) {	
overlay_idx[i]	ue(v)
language_overlay_present_flag[i]	u(1)
overlay_content_layer_id[i]	u(6)
overlay_label_present_flag[i]	u(1)
if(overlay_label_present_flag[i])	
overlay_label_layer_id[i]	u(6)
overlay_alpha_present_flag[i]	u(1)
if(overlay_alpha_present_flag[i])	
overlay_alpha_layer_id[i]	u(6)
if(overlay_label_present_flag[i]) {	
num_overlay_elements_minus1[i]	ue(v)
for(j = 0; j <= num_overlay_elements_minus1[i]; j++) {	
overlay_element_label_min[i][j]	u(v)
overlay_element_label_max[i][j]	u(v)
}	
}	
}	
while(!byte_aligned( ))	
overlay_zero_bit /* equal to 0 */	f(1)
for(i = 0; i <= num_overlays_minus1; i++) {	
if(language_overlay_present_flag[i])	
overlay_language[i]	st(v)
overlay_name[i]	st(v)
if(overlay_label_present_flag[i])	
for(j = 0; j <= num_overlay_elements_minus1[i]; j++)	
overlay_element_name[i][j]	st(v)
}	
overlay_info_persistence_flag	u(1)
}	
}	

F.14.2.10 Temporal motion vector prediction constraints SEI message syntax

temporal_mv_prediction_constraints(payloadSize) {	**Descriptor**
prev_pics_not_used_flag	u(1)
no_intra_layer_col_pic_flag	u(1)
}	

F.14.2.11 Frame-field information SEI message syntax

frame_field_info(payloadSize) {	**Descriptor**
ffinfo_pic_struct	u(4)
ffinfo_source_scan_type	u(2)
ffinfo_duplicate_flag	u(1)
}	

F.14.3 SEI payload semantics

F.14.3.1 General SEI payload semantics

The general SEI payload semantics specified in clause D.3.1 apply with the following modifications and additions:

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 161, 165, 167, 168, 200 to 202, inclusive, and 205.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 168, inclusive, 200 to 202, inclusive, and 205.

The semantics and persistence scope for each SEI message specified in this annex are specified in the semantics specification for each particular SEI message in clauses F.14.3.3 to F.14.3.11, inclusive.

NOTE – Persistence information for SEI messages specified in this annex is informatively summarized in Table F.4.

Table F.4 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Layers not present	Specified by the semantics of the SEI message
Inter-layer constrained tile sets	The CLVS associated with the SEI message
Bitstream partition nesting	Each SEI message contained in the bitstream partition nesting SEI message has the same persistence scope as if the SEI message was not contained in the bitstream partition nesting SEI message
Bitstream partition initial arrival time	The remainder of the bitstream partition (specified by the containing bitstream partition nesting SEI message)
Sub-bitstream property	The CVS containing the SEI message
Alpha channel information	Specified by the syntax of the SEI message
Overlay information	Specified by the syntax of the SEI message
Temporal motion vector prediction constraints	Specified by the semantics of the SEI message
Frame-field information	The access unit containing the SEI message

Let prevVclNalUnitInAu of an SEI NAL unit or an SEI message be the preceding VCL NAL unit in decoding order, if any, in the same access unit, and nextVclNalUnitInAu of an SEI NAL unit or an SEI message be the next VCL NAL unit in decoding order, if any, in the same access unit. It is a requirement of bitstream conformance that the following restrictions apply:

– When a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition initial arrival time SEI message is present in a bitstream partition nesting SEI message contained in a scalable nesting SEI message, the scalable nesting SEI message shall not follow any other SEI message that

follows the prevVclNalUnitInAu of the scalable nesting SEI message and precedes the nextVclNalUnitInAu of the scalable nesting SEI message, other than an active parameter sets SEI message, a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, a non-scalable-nested decoding unit information SEI message, a scalable nesting SEI message including a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or another scalable nesting SEI message that contains a bitstream partition nesting SEI message including a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition initial arrival time SEI message.

F.14.3.2 Annex D SEI message semantics for multi-layer extensions

F.14.3.2.1 General

The semantics of SEI messages specified in clauses D.3.2 through D.3.49 apply with the modifications specified in clauses F.14.3.2.2 through F.14.3.2.8.

F.14.3.2.2 Buffering period SEI message semantics for multi-layer extensions

The specifications of clause D.3.2 apply with the following modifications and additions:

When the buffering period SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with a nuh_layer_id value targetNuhLayerId greater than 0, the semantics of clause D.3.2 apply to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables assignedBaseLayerId equal to nestingLayerIdList[i][0] and tIdTarget equal to MaxTemporalId[i] for each value of i in the range of 0 to nestingNumOps − 1, inclusive.

When the buffering period SEI message is contained in a bitstream partition nesting SEI message, the following applies:

– A set of skipped leading pictures skippedPictureList consists of the CL-RAS pictures and the RASL pictures associated with the IRAP pictures with nuh_layer_id equal to nuhLayerId for which LayerInitializedFlag[nuhLayerId] is equal to 0 at the start of decoding the IRAP picture and for which nuhLayerId is among the nestingLayerIdList[i] for any value of i in the range of 0 to nesting_num_ops_minus1, inclusive.

– A picture is said to be a notDiscardablePic picture when the picture has TemporalId equal to 0, has discardable_flag equal to 0, and is not a RASL, RADL or SLNR picture.

– The variables olsIdx, psIdx and bspIdx are set equal to sei_ols_idx, sei_partitioning_scheme_idx and bsp_idx, respectively, of the bitstream partition nesting SEI message containing this buffering period SEI message and the bspIdx-th bitstream partition of the psIdx-th partitioning scheme of the olsIdx-th OLS is referred to as the current bitstream partition.

– The variable maxTId is set equal to MaxTemporalId[0] for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this buffering period SEI message.

– The variable CpbCnt is set equal to BspSchedCnt[olsIdx][psIdx][maxTId] for the syntax and semantics of the buffering period SEI message and for the constraints specified below.

– The variable hrdParamIdx[i] is set equal to the value of bsp_hrd_idx[olsIdx][psIdx][maxTId][i][bspIdx] for each value of i in the range of 0 to CpbCnt − 1, inclusive.

– The syntax elements au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1 and sub_pic_hrd_params_present_flag are found in or derived from the hrdParamIdx[i]-th hrd_parameters( ) syntax structure for any value of i in the range of 0 to CpbCnt − 1, inclusive. It is a requirement of bitstream conformance that the values of au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1 and sub_pic_hrd_params_present_flag derived using a particular value of i shall be the same as the value of the respective syntax elements derived using any other allowed value of i.

– The syntax element initial_cpb_removal_delay_length_minus1 is found in or derived from the hrdParamIdx[i]-th hrd_parameters( ) syntax structure for the semantics of nal_initial_cpb_removal_delay[i], nal_initial_alt_cpb_removal_delay[i], nal_initial_cpb_removal_offset[i], nal_initial_alt_cpb_removal_offset[i], vcl_initial_cpb_removal_delay[i], vcl_initial_alt_cpb_removal_delay[i], vcl_initial_cpb_removal_offset[i] and vcl_initial_alt_cpb_removal_offset[i].

– The variable CpbSize[i] is set equal to BpbSize[olsIdx][psIdx][maxTId][i][bspIdx] for the semantics of nal_initial_cpb_removal_delay[i], nal_initial_alt_cpb_removal_delay[i], vcl_initial_cpb_removal_delay[i] and vcl_initial_alt_cpb_removal_delay[i].

– The variable BitRate[i] is set equal to BpBitRate[olsIdx][psIdx][maxTId][i][bspIdx] for the semantics of nal_initial_cpb_removal_delay[i], nal_initial_alt_cpb_removal_delay[i], vcl_initial_cpb_removal_delay[i] and vcl_initial_alt_cpb_removal_delay[i].

– The variables NalHrdBpPresentFlag and VclHrdBpPresentFlag are derived from the hrdParamIdx[i]-th hrd_parameters( ) syntax structure for any value of i in the range of 0 to CpbCnt − 1, inclusive. It is a requirement of bitstream conformance that the value of NalHrdBpPresentFlag shall be the same regardless of the value of i used in its derivation. It is a requirement of bitstream conformance that the value of VclHrdBpPresentFlag shall be the same regardless of the value of i used in its derivation.

The presence of buffering period SEI messages for the current bitstream partition is specified as follows on the basis of the variables NalHrdBpPresentFlag and VclHrdBpPresentFlag that are derived as specified above:

– If NalHrdBpPresentFlag is equal to 1 or VclHrdBpPresentFlag is equal to 1, the following applies for each access unit in the CVS:
  – If the access unit is an IRAP access unit, a buffering period SEI message applicable to the current bitstream partition shall be associated with the access unit.
  – Otherwise, if the access unit contains a notDiscardablePic in at least one layer included in the current bitstream partition, a buffering period SEI message applicable to the current bitstream partition may or may not be associated with the access unit.
  – Otherwise, the access unit shall not be associated with a buffering period SEI message applicable to the current bitstream partition.
– Otherwise (NalHrdBpPresentFlag and VclHrdBpPresentFlag are both equal to 0), no access unit in the CVS shall be associated with a buffering period SEI message applicable to the current bitstream partition.

When the buffering period SEI message is contained in a bitstream partition nesting SEI message, the following semantics of concatenation_flag and au_cpb_removal_delay_delta_minus1 replace those specified in clause D.3.2 and the following specifications apply to each picture currPic with nuh_layed_id currNuhLayerId equal to any value included in the current bitstream partition in the current access unit.

When the picture currPic with nuh_layer_id equal to targetLayerId in the current access unit is not the first picture with that nuh_layer_id value in the bitstream in decoding order, let prevNonDiscardablePic be the preceding picture in decoding order with that nuh_layer_id value and with TemporalId equal to 0 that is not a RASL, RADL or SLNR picture.

concatenation_flag indicates, when the picture currPic is not the first picture with nuh_layer_id equal to currNuhLayerId in the bitstream in decoding order, whether the nominal CPB removal time of the current picture is determined relative to the nominal CPB removal time of the preceding picture with a buffering period SEI message that applies to the current bitstream partition, or relative to the nominal CPB removal time of the picture prevNonDiscardablePic.

au_cpb_removal_delay_delta_minus1 plus 1, when the picture currPic is not the first picture with nuh_layer_id equal to currNuhLayerId in the bitstream in decoding order, specifies a CPB removal delay increment value relative to the nominal CPB removal time of the picture prevNonDiscardablePic. This syntax element has a length in bits given by au_cpb_removal_delay_length_minus1 + 1.

When the buffering period SEI message is contained in a bitstream partition nesting SEI message and concatenation_flag is equal to 0 and the picture currPic is not the first picture with nuh_layer_id equal to currNuhLayerId in the bitstream in decoding order, it is a requirement of bitstream conformance that the following constraint applies:

  – If the picture prevNonDiscardablePic is not associated with a buffering period SEI message that applies the current bitstream partition, the au_cpb_removal_delay_minus1 of the picture currPic shall be equal to the au_cpb_removal_delay_minus1 of prevNonDiscardablePic, indicated in the picture timing SEI message applicable to the current bitstream partition, plus au_cpb_removal_delay_delta_minus1 + 1.

– Otherwise, au_cpb_removal_delay_minus1, indicated for the picture currPic in the picture timing SEI message applicable to the current bitstream partition, shall be equal to au_cpb_removal_delay_delta_minus1.

F.14.3.2.3 Picture timing SEI message semantics for multi-layer extensions

The specifications of clause D.3.3 apply with the following modifications and additions:

When the picture timing SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with a nuh_layer_id value targetNuhLayerId greater than 0, the semantics of clause D.3.3 apply to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables assignedBaseLayerId equal to nestingLayerIdList[i][0] and tIdTarget equal to MaxTemporalId[i] for each value of i in the range of 0 to nestingNumOps − 1, inclusive.

When the picture timing SEI message is contained in a bitstream partition nesting SEI message, the following applies:

– frame_field_info_present_flag is inferred to be equal to 0 for the syntax and semantics of the picture timing SEI message.

NOTE 1 – The frame-field information SEI message can be used to indicate the frame-field information.

– The variables olsIdx, psIdx and bspIdx are set equal to sei_ols_idx, sei_partitioning_scheme_idx and bsp_idx, respectively, of the bitstream partition nesting SEI message containing this buffering period SEI message, and the bspIdx-th bitstream partition of the psIdx-th partitioning scheme of the olsIdx-th OLS is referred to as the current bitstream partition.

– The variable maxTId is set equal to MaxTemporalId[0] for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this buffering period SEI message.

– The variable hrdParamIdx is set equal to the value of bsp_hrd_idx[olsIdx][psIdx][maxTId][i][bspIdx] for any value of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][maxTId] − 1, inclusive.

– The following applies for the syntax and semantics of the picture timing SEI message:

  – The syntax elements sub_pic_hrd_params_present_flag, sub_pic_cpb_params_in_pic_timing_sei_flag, au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1, dpb_output_delay_du_length_minus1, du_cpb_removal_delay_increment_length_minus1 and the variable CpbDpbDelaysPresentFlag are found in or derived from syntax elements found in the hrdParamIdx-th hrd_parameters( ) syntax structure.

  – It is a requirement of bitstream conformance that the values of sub_pic_hrd_params_present_flag, sub_pic_cpb_params_in_pic_timing_sei_flag, au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1, dpb_output_delay_du_length_minus1, du_cpb_removal_delay_increment_length_minus1 and CpbDpbDelaysPresentFlag derived using a particular value of i for the derivation of hrdParamIdx shall be the same as the value of the respective syntax elements or variable derived using any other allowed value of i.

The presence of picture timing SEI messages for the current bitstream partition is specified as follows on the basis of the variable CpbDpbDelaysPresentFlag that is derived as specified above:

– If CpbDpbDelaysPresentFlag is equal to 1, a picture timing SEI message applicable to the current bitstream partition shall be associated with every access unit in the CVS.

– Otherwise, in the CVS there shall be no access unit that is associated with a picture timing SEI message applicable to the current bitstream partition.

The semantics of num_decoding_units_minus1 and num_nalus_in_du_minus1[i] are replaced with the following:

The variables targetUnit and totalSizeInCtbsY are derived as follows:

– If the picture timing SEI message is associated with a partition unit, targetUnit is set to be the partition unit and the value of totalSizeInCtbsY is set equal to the sum of the PicSizeInCtbsY of all pictures in the partition unit.

– Otherwise (the picture timing SEI message is associated with an access unit), targetUnit is set to be the access unit and the value of totalSizeInCtbsY is set equal to the sum of the PicSizeInCtbsY of all pictures in the access unit.

num_decoding_units_minus1 plus 1 specifies the number of decoding units in targetUnit. The value of num_decoding_units_minus1 shall be in the range of 0 to totalSizeInCtbsY − 1, inclusive.

num_nalus_in_du_minus1[i] plus 1 specifies the number of NAL units in the i-th decoding unit of targetUnit. The value of num_nalus_in_du_minus1[i] shall be in the range of 0 to totalSizeInCtbsY − 1, inclusive.

The first decoding unit of targetUnit consists of the first num_nalus_in_du_minus1[0] + 1 consecutive NAL units in decoding order in targetUnit. The i-th (with i greater than 0) decoding unit of targetUnit consists of the num_nalus_in_du_minus1[i] + 1 consecutive NAL units immediately following the last NAL unit in the previous decoding unit of targetUnit, in decoding order. There shall be at least one VCL NAL unit in each decoding unit. All non-VCL NAL units associated with a VCL NAL unit shall be included in the same decoding unit as the VCL NAL unit.

F.14.3.2.4 Recovery point SEI message semantics for multi-layer extensions

The recovery point SEI message assists a decoder in determining when the decoding process will produce acceptable pictures in the current layer for display after the decoder initiates random access, layer up-switching or after the encoder indicates a broken link.

When all decoded pictures in earlier access units in decoding order are removed from the bitstream, the recovery point picture (defined below) and all the subsequent pictures in output order in the current layer can be correctly or approximately correctly decoded, the current picture is referred to as a layer random-accessing picture. When all the pictures that belong to the reference layers of the current layer and may be used for reference by the current picture or subsequent pictures in decoding order are correctly decoded, and the recovery point picture and all the subsequent pictures in output order in the current layer can be correctly or approximately correctly decoded when no picture prior to the current picture in decoding order in the current layer are present in the bitstream, the current picture is referred to as a layer up-switching picture.

When the recovery point SEI message applies to the current layer and all the reference layers of the current layer, the current picture is indicated as a layer random-accessing picture. When the recovery point SEI message applies to the current layer but not to all the reference layers of the current layer, the current picture is indicated as a layer up-switching picture.

Decoded pictures in the current layer produced by random access or layer up-switching at or before the current access unit does not need to be correct in content until the indicated recovery point, and the operation of the decoding process for pictures in the current layer starting at the current picture may contain references to pictures unavailable in the decoded picture buffer.

In addition, by use of the broken_link_flag, the recovery point SEI message can indicate to the decoder the location of some pictures in the current layer in the bitstream that can result in serious visual artefacts when displayed, even when the decoding process was begun at the location of a previous IRAP access unit in decoding order that contain IRAP pictures in all layers.

NOTE 1 – The broken_link_flag can be used by encoders to indicate the location of a point after which the decoding process for the decoding of some pictures in the current layer may cause references to pictures that, though available for use in the decoding process, are not the pictures that were used for reference when the bitstream was originally encoded (e.g., due to a splicing operation performed during the generation of the bitstream).

When the current picture is a layer random access-accessing picture and random access is performed to start decoding from the current access unit, the decoder operates as if the current access unit was the first access unit in the bitstream in decoding order, and the following applies:

– If poc_msb_cycle_val is present for the current picture, the PicOrderCntVal of the current picture is derived as specified in the process for derivation of PicOrderCntVal in clause F.8.3.1.

– Otherwise (poc_msb_cycle_val is not present for the current picture), the variable PrevPicOrderCnt[nuh_layer_id] used in derivation of PicOrderCntVal for each picture in the access unit is set equal to 0.

NOTE 2 – When HRD information is present in the bitstream, a buffering period SEI message should be associated with the access unit associated with the recovery point SEI message in order to establish initialization of the HRD buffer model after a random access.

When the current picture is either a layer random-accessing picture or a layer up-switching picture and layer up-switching is performed to start decoding of the current layer from the current access (while decoding of the reference layers of the current layer have started earlier and pictures of those layers in the current access unit are correctly decoded), the decoder operates as if the current picture was the first picture of the current layer in the bitstream in decoding order, and the PicOrderCntVal of the current picture is set equal to the PicOrderCntVal of any other decoded picture in the current access unit.

For a particular access unit auA, let auB be the first access unit that succeeds auA in decoding order such that the picture order count values of the pictures in auB can be derived without using the picture order count values of pictures preceding auB in decoding order. For a particular layer layerA, auA is not allowed to contain a recovery point SEI message that applies to a set of layers containing at least layerA and all reference layers of layerA when all of the following conditions are true:

– All pictures in auA that are in the reference layers, if any, of layerA have both poc_msb_cycle_val_present_flag and poc_reset_idc equal to 0.

– There is at least one picture picA with poc_msb_cycle_val_present_flag equal to 1 in the following access units:

  – Access units that follow auA in decoding order and precede auB, when present, in decoding order.

  – Access unit auB, when present and when picture in auB with nuh_layer_id equal to 0 is not an IRAP picture with NoClrasOutputFlag equal to 1.

Any SPS or PPS RBSP that is referred to by a picture of the access unit containing a recovery point SEI message or by any picture in a subsequent access unit in decoding order shall be available to the decoding process prior to its activation, regardless of whether or not the decoding process is started at the beginning of the bitstream or with the access unit, in decoding order, that contains the recovery point SEI message.

recovery_poc_cnt specifies the recovery point of decoded pictures in the current layer in output order. If there is a picture picB in the current layer that follows the current picture picA but precedes an access unit containing an IRAP picture in the current layer in decoding order and PicOrderCnt(picB) is equal to PicOrderCnt(picA) plus the value of recovery_poc_cnt, where PicOrderCnt(picA) and PicOrderCnt(picB) are the PicOrderCntVal values of picA and picB, respectively, immediately after the invocation of the decoding process for picture order count for picB, the picture picB is referred to as the recovery point picture. Otherwise, the first picture picC in the current layer in output order for which PicOrderCnt(picC) is greater than PicOrderCnt(picA) plus the value of recovery_poc_cnt is referred to as the recovery point picture, where PicOrderCnt(picA) and PicOrderCnt(picC) are the PicOrderCntVal values of picA and picC, respectively, immediately after the invocation of the decoding process for picture order count for picC. The recovery point picture shall not precede the current picture in decoding order. All decoded pictures in the current layer in output order are

indicated to be correct or approximately correct in content starting at the output order position of the recovery point picture. The value of recovery_poc_cnt shall be in the range of −MaxPicOrderCntLsb / 2 to MaxPicOrderCntLsb / 2 − 1, inclusive.

exact_match_flag indicates whether decoded pictures in the current layer at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit containing the recovery point SEI message will be an exact match to the pictures in the current layer that would be produced by starting the decoding process at the location of a previous access unit where the picture of the layer in the current layer and the pictures of all the reference layers are IRAP pictures, if any, in the bitstream. The value 0 indicates that the match may not be exact and the value 1 indicates that the match will be exact. When exact_match_flag is equal to 1, it is a requirement of bitstream conformance that the decoded pictures in the current layer at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit containing the recovery point SEI message shall be an exact match to the pictures in the current layer that would be produced by starting the decoding process at the location of a previous access unit where the picture in the current layer and the pictures of all the reference layers are IRAP pictures, if any, in the bitstream.

NOTE 3 – When performing random access, decoders should infer all references to unavailable pictures as references to pictures containing only intra coding blocks and having sample values given by Y equal to (1 << ($BitDepth_Y$ − 1)), Cb and Cr both equal to (1 << ($BitDepth_C$ − 1)) (mid-level grey), regardless of the value of exact_match_flag.

When exact_match_flag is equal to 0, the quality of the approximation at the recovery point is chosen by the encoding process and is not specified in this Specification.

broken_link_flag indicates the presence or absence of a broken link in the current layer at the location of the recovery point SEI message and is assigned further semantics as follows:

– If broken_link_flag is equal to 1, pictures in the current layer produced by starting the decoding process at the location of a previous access unit where the picture in the current layer and the pictures of all the reference layers are IRAP pictures may contain undesirable visual artefacts to the extent that decoded pictures in the current layer at and subsequent to the access unit containing the recovery point SEI message in decoding order should not be displayed until the specified recovery point in output order.
– Otherwise (broken_link_flag is equal to 0), no indication is given regarding any potential presence of visual artefacts.

When the current picture is a BLA picture, the value of broken_link_flag shall be equal to 1.

Regardless of the value of the broken_link_flag, pictures in the current layer subsequent to the specified recovery point in output order are specified to be correct or approximately correct in content.

F.14.3.2.5 Structure of pictures information SEI message semantics for multi-layer extensions

The specifications of clause D.3.19 apply by replacing the first paragraph specifying the persistence of the SEI message with the following:

The structure of pictures information SEI message provides information for a list of entries, some of which correspond to the target picture set consists of a series of pictures starting from the current picture until the last picture in decoding order in the current layer in the CLVS or the last picture in decoding order in the current POC resetting period, whichever is earlier.

F.14.3.2.6 Decoding unit information SEI message semantics for multi-layer extensions

The specifications of clause D.3.22 apply with the following modifications and additions:

When the decoding unit information SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with a nuh_layer_id value targetNuhLayerId greater than 0, the semantics of clause D.3.22 apply to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables assignedBaseLayerId equal to nestingLayerIdList[i][0] and tIdTarget equal to MaxTemporalId[i] for each value of i in the range of 0 to nestingNumOps − 1, inclusive.

When the decoding unit information SEI message is contained in a bitstream partition nesting SEI message, the following applies:

– The variables olsIdx, psIdx and bspIdx are set equal to sei_ols_idx, sei_partitioning_scheme_idx and bsp_idx, respectively, of the bitstream partition nesting SEI message containing this buffering period SEI message, and the bspIdx-th bitstream partition of the psIdx-th partitioning scheme of the olsIdx-th OLS is referred to as the current bitstream partition.
– The variable maxTId is set equal to MaxTemporalId[0] for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this buffering period SEI message.
– The variable hrdParamIdx is set equal to the value of bsp_hrd_idx[olsIdx][psIdx][maxTId][i][bspIdx] for any value of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][maxTId] − 1, inclusive.

– The following applies for the syntax and semantics of the decoding unit information SEI message:

  – The syntax elements sub_pic_hrd_params_present_flag, sub_pic_cpb_params_in_pic_timing_sei_flag and dpb_output_delay_du_length_minus1, and the variable CpbDpbDelaysPresentFlag are found in or derived from syntax elements in the hrdParamIdx-th hrd_parameters( ) syntax structure.

  – It is a requirement of bitstream conformance that the values of sub_pic_hrd_params_present_flag, sub_pic_cpb_params_in_pic_timing_sei_flag and dpb_output_delay_du_length_minus1, and CpbDpbDelaysPresentFlag derived using a particular value of i for the derivation of hrdParamIdx shall be the same as the value of the respective syntax elements or variable derived using any other allowed value of i.

The presence of decoding unit information SEI messages for the current bitstream partition is specified as follows on the basis of the syntax elements sub_pic_hrd_params_present_flag and sub_pic_cpb_params_in_pic_timing_sei_flag and the variable CpbDpbDelaysPresentFlag that are found or derived as specified above:

– If CpbDpbDelaysPresentFlag is equal to 1, sub_pic_hrd_params_present_flag is equal to 1 and sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 0, one or more decoding unit information SEI messages applicable to the current bitstream partition shall be associated with each decoding unit of the current bitstream partition in the CVS.

– Otherwise, if CpbDpbDelaysPresentFlag is equal to 1, sub_pic_hrd_params_present_flag is equal to 1 and sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 1, one or more decoding unit information SEI messages applicable to the current bitstream partition may or may not be associated with each decoding unit of the current bitstream partition in the CVS.

– Otherwise (CpbDpbDelaysPresentFlag is equal to 0 or sub_pic_hrd_params_present_flag is equal to 0), in the CVS there shall be no decoding unit of the current bitstream partition that is associated with a decoding unit information SEI message applicable to the current bitstream partition.

The set of NAL units associated with a decoding unit information SEI message contained in a bitstream partition nesting SEI message consists, in decoding order, of the SEI NAL unit containing the decoding unit information SEI message and all subsequent NAL units in the partition unit up to but not including any subsequent SEI NAL unit containing a decoding unit information SEI message that is contained in a bitstream partition nesting SEI message applying to the current bitstream partition and that has a different value of decoding_unit_idx.

It is a requirement of bitstream conformance that all decoding unit information SEI messages that are contained in a bitstream partition nesting SEI message, are associated with the same access unit, apply to the current bitstream partition and have dpb_output_du_delay_present_flag equal to 1 shall have the same value of pic_spt_dpb_output_du_delay.

F.14.3.2.7 Scalable nesting SEI message semantics for multi-layer extensions

The specifications of clause D.3.24 apply with the following modifications:

It is a requirement of bitstream conformance that the following restrictions apply on containing of SEI messages in a scalable nesting SEI message:

– An SEI message that has payloadType equal to 129 (active parameter sets), 132 (decoded picture hash), 133 (scalable nesting), 160 (layers not present), 163 (bitstream partition initial arrival time), 164 (sub-bitstream property) or 166 (overlay information) shall not be directly contained in a scalable nesting SEI message.

– When a scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, the scalable nesting SEI message shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information).

– When present, an SEI message that has payloadType equal to 162 (bitstream partition nesting) shall be contained within a scalable nesting SEI message.

– When a scalable nesting SEI message contains a bitstream partition nesting SEI message, the scalable nesting SEI message shall not contain any other SEI message.

It is a requirement of bitstream conformance that the following restrictions apply on the value of bitstream_subset_flag:

– When the scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition nesting SEI message, bitstream_subset_flag shall be equal to 1.

– When the scalable nesting SEI message contains an SEI message that has payloadType equal to any value among SingleLayerSeiList, 161, 165, 167 or 168, bitstream_subset_flag shall be equal to 0.

When nesting_op_idx[i] is present, the list nestingLayerIdList[i] is set equal to LayerSetLayerIdList[nesting_op_idx[i]] derived from the active VPS.

When a scalable nesting SEI message contains a bitstream partition nesting SEI message, the following applies:

– nesting_op_flag shall be equal to 1, default_op_flag shall be equal to 0, nesting_num_ops_minus1 shall be equal to 0 and nesting_op_idx[0] shall not be equal to 0.

– The nuh_layer_id of the SEI NAL unit containing the scalable nesting SEI message shall be equal to the highest value within the list nestingLayerIdList[0].

When a scalable nesting SEI message directly contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message and the nuh_layer_id value of the SEI NAL unit containing the scalable nesting SEI messge is greater than 0, it is a requirement of bitstream conformance that for any value of i in the range of default_op_flag to nesting_num_ops_minus1, inclusive, the nesting_op_idx[i]-th OLS consists of only the layer with nuh_layer_id equal to nestingLayerIdList[i][0], the profile of that layer is a profile specified in Annex A and the value of general_inbld_flag (when MaxTemporalId[i] is equal to vps_max_sub_layers_minus1) or the value of sub_layer_inbld_flag[tidTarget] (when MaxTemporalId[i] is less than vps_max_sub_layers_minus1) in the profile_tier_level( ) syntax structure associated with that layer is equal to 1.

NOTE – In other words, buffering period, picture timing, or decoding unit information SEI messages that are directly contained in an SEI NAL unit with nuh_layer_id greater than 0 can only be present for rewritable independent non-base layers.

F.14.3.2.8 Region refresh information SEI message semantics for multi-layer extensions

The region refresh information SEI message indicates whether the slice segments that the current SEI message applies to belong to a refreshed region of the current picture.

The variable targetLayerIdList is derived as follows:

– If the region refresh information SEI message applies to the current layer and all the reference layers, targetLayerIdList contains the nuh_layer_id of the current layer and all the reference layers.

– Otherwise, targetLayerIdList contains the nuh_layer_id of the current layer.

The region refresh SEI message is associated with a recovery point SEI message that applies to targetLayerIdList.

A picture that is not an IRAP picture, that belongs to a layer, and that is contained in an access unit containing a recovery point SEI message where the recovery point SEI message applies to that layer is referred to as a gradual decoding refresh (GDR) picture, and the access unit containing the picture is referred to as a GDR access unit. The access unit corresponding to the indicated recovery point picture is referred to as the recovery point access unit.

If there is a picture picB in the current layer that follows the GDR picture picA in the current layer in decoding order in the CVS and PicOrderCnt(picB) is equal to PicOrderCnt(picA) plus the value of recovery_poc_cnt in the recovery point SEI message, where PicOrderCnt(picA) and PicOrderCnt(picB) are the PicOrderCntVal values of picA and picB, respectively, immediately after the invocation of the decoding process for picture order count for picB, let the variable lastPicInSet be the recovery point picture. Otherwise, let lastPicInSet be the picture in targetLayerIdList that immediately precedes the recovery point picture in output order. The picture lastPicInSet shall not precede the GDR access unit in decoding order.

Let gdrPicSet be the set of pictures in targetLayerIdList starting from a GDR access unit to the access unit containing lastPicInSet, inclusive, in output order. When the decoding process for the current layer is started from a GDR access unit, the refreshed region in each picture of the gdrPicSet is indicated to be the region of the picture that is correct or approximately correct in content, and, when lastPicInSet is contained in the recovery point access unit, the refreshed region in lastPicInSet covers the entire picture.

The slice segments of the current picture to which a region refresh information SEI message applies consist of all slice segments within the picture that follow the SEI NAL unit containing the region refresh information SEI message and precede the next SEI NAL unit, in decoding order, containing a region refresh information SEI message (if any) that has the same targetLayerIdList as the current SEI message. These slice segments are referred to as the slice segments associated with the region refresh information SEI message.

Let gdrAuSet be the set of access units corresponding to gdrPicSet. A gdrAuSet and the corresponding gdrPicSet are referred to as being associated with the recovery point SEI message contained in the GDR access unit.

Region refresh information SEI messages shall not be present in an access unit unless the access unit is included in a gdrAuSet associated with a recovery point SEI message. When any picture that is included in a gdrPicSet is associated with one or more region refresh information SEI messages, all pictures in the gdrPicSet shall be associated with one or more region refresh information SEI messages.

refreshed_region_flag equal to 1 indicates that the slice segments associated with the current SEI message belong to the refreshed region in the current picture. refreshed_region_flag equal to 0 indicates that the slice segments associated with the current SEI message may not belong to the refreshed region in the current picture.

When one or more region refresh information SEI messages are associated with a picture belonging to gdrPicSet and the first slice segment of the picture in decoding order does not have an associated region refresh information SEI message, the value of refreshed_region_flag for the slice segments of the picture that precede the first region refresh information SEI message is inferred to be equal to 0.

When lastPicInSet is the recovery point picture, and any region refresh SEI message is associated with the recovery point access unit, the first slice segment of the picture in decoding order shall have an associated region refresh SEI message, and the value of refreshed_region_flag shall be equal to 1 in all region refresh SEI messages associated with the picture.

When one or more region refresh information SEI messages are associated with a picture, the refreshed region in the picture is specified as the set of CTUs in all slice segments of the picture that are associated with region refresh information SEI messages that have refreshed_region_flag equal to 1. Other slice segments belong to the non-refreshed region of the picture.

It is a requirement of bitstream conformance that when a dependent slice segment belongs to the refreshed region, the preceding slice segment in decoding order shall also belong to the refreshed region.

Let gdrRefreshedSliceSegmentSet be the set of all slice segments that belong to the refreshed regions in the gdrPicSet. The variable upSwitchingRefreshedSliceSegmentSet is derived as follows:

– If targetLayerIdList contains only one non-zero nuh_layer_id, upSwitchingRefreshedSliceSegmentSet is defined as the set inclusive of the following:
  – all slice segments of all pictures of the reference layers that precede, in decoding order, the current picture and that may be used for reference by the current picture or subsequent pictures of the reference layers.
  – all slice segments of all pictures of the reference layers that succeed, in decoding order, the current picture and that belong to gdrAuSet.
– Otherwise, upSwitchingRefreshedSliceSegmentSet is defined as an empty set.

When a gdrPicSet contains one or more pictures associated with region refresh information SEI messages, it is a requirement of bitstream conformance that the following constraints all apply:

– For each layer in targetLayerIdList, the refreshed region in the first picture, in decoding order, that belongs to the layer and that is included in gdrPicSet that contains any refreshed region shall contain only coding units that are coded in an intra coding mode or inter-layer prediction from slice segments belonging to the union of gdrRefreshedSliceSegmentSet and upSwitchingRefreshedSliceSegmentSet.
– For each picture included in the gdrPicSet, the syntax elements in gdrRefreshedSliceSegmentSet shall be constrained such that no samples or motion vector values outside of the union of gdrRefreshedSliceSegmentSet and upSwitchingRefreshedSliceSegmentSet are used for inter prediction or inter-layer prediction in the decoding process of any samples within gdrRefreshedSliceSegmentSet.
– For any picture that follows the picture lastPicInSet in output order, the syntax elements in the slice segments of the picture shall be constrained such that no samples or motion vector values outside of the union of gdrRefreshedSliceSegmentSet and upSwitchingRefreshedSliceSegmentSet are used for inter prediction or inter-layer prediction in the decoding process of the picture other than those of the other pictures that follow the picture lastPicInSet in output order.

F.14.3.2.9 Coded region completion SEI message semantics for multi-layer extensions

The specifications of clause D.3.37 apply with the following additions:

The nuh_layer_id value of the SEI NAL unit containing a coded region completion SEI message shall be equal to the nuh_layer_id value of the associated VCL NAL unit for the SEI NAL unit.

When an access unit contains one or more layers not present SEI messages, any coded region completion SEI message with next_segment_address equal to 0 in that access unit shall follow the first layers not present SEI message, in decoding order, that is present in that access unit.

NOTE – The specifications given in clause F.7.4.2.4.4 for associating NAL units to access units facilitate determining of an end of an access unit when the first NAL unit of the next access unit is available. Some implementations may operate on access unit basis, e.g. when inputting NAL units to a decoder. The latency in such implementations is potentially reduced, when a bitstream includes multiple layers and an end of an access unit for the OLS with index trgtOlsIdx is concluded as follows in response to decoding a coded region completion SEI message with next_segment_address equal to 0:

– The variable layerMayBePresent[layerIdx] is derived as follows:
  – If a layers not present SEI message is present in the current CVS and precedes the coded region completion SEI message in decoding order, layerMayBePresentFlag[layerIdx] is set equal to !layer_not_present_flag[layerIdx] of the previous layers not present SEI message, in decoding order, for each value of layerIdx from 0 to MaxLayersMinus1, inclusive.
  – Otherwise, layerMayBePresentFlag[layerIdx] is set equal to 1 for each value of layerIdx from 0 to MaxLayersMinus1, inclusive.

- Let a set of nuh_layer_id values trgtLayerIdList of the necessary layers for an OLS with index trgtOlsIdx be equal to TargetDecLayerIdList derived using Equation F-57 by setting TargetOlsIdx equal to trgtOlsIdx.
- Let numTrgtLayerIds be equal to the number of nuh_layer_id values in trgtLayerIdList.
- Let currLayerId be equal to the nuh_layer_id value of the VCL NAL unit associated with the SEI NAL unit containing this coded region completion SEI message.
- Let prevNecessaryLayerTrgtIdx be the index of the greatest value within trgtLayerIdList that is less than or equal to currLayerId.
- When one of the following is true, there are no VCL NAL units of any necessary layer of the OLS with index trgtOlsIdx following, in decoding order, the VCL NAL unit associated with the SEI NAL unit containing this SEI message within the current access unit:
  - currLayerId is greater than or equal to any value in trgtLayerIdList.
  - layerMayBePresentFlag[LayerIdxInVps[layerId]] is equal to 0 for all the values of layerId equal to trgtLayerIdList[trgtIdx] for each trgtIdx in the range of prevNecessaryLayerTrgtIdx + 1 to numTrgtLayerIds − 1, inclusive.

F.14.3.3 Layers not present SEI message semantics

The layers not present SEI message provides a mechanism for signalling that VCL NAL units of particular layers indicated by the VPS are not present in a particular set of access units.

The target access units are defined as the set of access units starting from the access unit containing the layers not present SEI message up to but not including the next access unit, in decoding order, that contains a layers not present SEI message or the end of the CVS, whichever is earlier in decoding order.

When present, the layers not present SEI message applies to the target access units.

When a layers not present SEI message is associated with a coded picture with TemporalId firstTid that is greater than 0 and that coded picture is followed, in decoding order, by any coded picture with TemporalId less than firstTid in the same CVS, a layers not present SEI message shall be present for the next coded picture, in decoding order, with TemporalId less than firstTid.

lnp_sei_active_vps_id identifies the active VPS of the CVS containing the layers not present SEI message. The value of lnp_sei_active_vps_id shall be equal to the value of vps_video_parameter_set_id of the active VPS for the VCL NAL units of the access unit containing the SEI message.

layer_not_present_flag[i] equal to 1 indicates that there are no VCL NAL units with nuh_layer_id equal to layer_id_in_nuh[i] present in the target access units. layer_not_present_flag[i] equal to 0 indicates that there may or may not be VCL NAL units with nuh_layer_id equal to layer_id_in_nuh[i] present in the target access units.

When layer_not_present_flag[i] is equal to 1 and i is less than MaxLayersMinus1, layer_not_present_flag[LayerIdxInVps[IdPredictedLayer[layer_id_in_nuh[i]][j]]] shall be equal to 1 for all values of j in the range of 0 to NumPredictedLayers[layer_id_in_nuh[i]] − 1, inclusive.

F.14.3.4 Inter-layer constrained tile sets SEI message semantics

The scope of the inter-layer constrained tile sets SEI message is a complete CLVS of the layer with nuh_layer_id equal to targetLayerId. When an inter-layer constrained tile sets SEI message is present for any coded picture of a CLVS and the first coded picture of the CLVS in decoding order is an IRAP picture, the inter-layer constrained tile sets SEI message shall be present for the first coded picture of the CLVS in decoding order and may also be present for other coded pictures of the CLVS.

The inter-layer constrained tile sets SEI message shall not be present for the layer with nuh_layer_id equal to targetLayerId when tiles_enabled_flag is equal to 0 for any PPS that is active for the pictures of the CLVS of the layer with nuh_layer_id equal to targetLayerId.

The inter-layer constrained tile sets SEI message shall not be present for the layer with nuh_layer_id equal to targetLayerId unless every PPS that is active for the pictures of the CLVS of the layer with nuh_layer_id equal to targetLayerId has tile_boundaries_aligned_flag equal to 1 or fulfills the conditions that would be indicated by tile_boundaries_aligned_flag being equal to 1.

An identified tile set is defined as a set of tiles within a picture with nuh_layer_id equal to targetLayerId that contains the tiles specified below on the basis of the values of the syntax elements contained in the inter-layer constrained tile sets SEI message.

An associated reference tile set for the iPred-th identified tile set with ilcts_id[iPred] equal to ilctsId, if any, is defined as the identified tile set present in the same access unit as the iPred-th identified tile set and associated with ilcts_id[iRef] equal to ilctsId and nuh_layer_id equal to IdDirectRefLayer[targetLayerId][j] for any value of j in the range of 0 to NumDirectRefLayers[targetLayerId] − 1, inclusive.

The presence of the inter-layer constrained tile sets SEI message indicates that the inter-layer prediction process is constrained such that no sample value outside each associated reference tile set, and no sample value at a fractional sample position that is derived using one or more sample values outside each associated reference tile set, is used for inter-layer prediction of any sample within the identified tile set.

NOTE 1 – When loop filtering and resampling filter are applied across tile boundaries, inter-layer prediction of any samples within an identified tile set that refers to samples within 8 samples from an associated reference tile set boundary that is not also a picture boundary may result in propagation of mismatch error. An encoder can avoid such potential error propagation by avoiding the use of motion vectors that cause such references.

When more than one inter-layer constrained tile sets SEI message is present for a CLVS, they shall contain identical content.

The number of inter-layer constrained tile sets SEI messages for the same CLVS in each access unit shall not exceed 5.

il_all_tiles_exact_sample_value_match_flag equal to 1 indicates that, within the CLVS, when the CTBs that are outside of any tile in any identified tile set specified in this inter-layer constrained tile sets SEI message are not decoded and the boundaries of the tile is treated as picture boundaries for purposes of the decoding process, the value of each sample in the tile would be exactly the same as the value of the sample that would be obtained when all the CTBs of all pictures in the CLVS are decoded. il_all_tiles_exact_sample_value_match_flag equal to 0 indicates that, within the CLVS, when the CTBs that are outside of any tile in any identified tile set specified in this inter-layer constrained tile sets SEI message are not decoded and the boundaries of the tile is treated as picture boundaries for purposes of the decoding process, the value of each sample in the tile may or may not be exactly the same as the value of the same sample when all the CTBs of all pictures in the CLVS are decoded.

il_one_tile_per_tile_set_flag equal to 1 indicates that each identified tile set contains one tile and il_num_sets_in_message_minus1 is not present.

It is a requirement of bitstream conformance that when il_one_tile_per_tile_set_flag is equal to 1, the value of (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) shall be less than or equal to 256.

When il_one_tile_per_tile_set_flag is equal to 0, the identified tile sets are signalled explicitly.

il_num_sets_in_message_minus1 plus 1 specifies the number of the identified tile sets in the SEI message. The value of il_num_sets_in_message_minus1 shall be in the range of 0 to 255, inclusive.

skipped_tile_set_present_flag equal to 1 indicates that, within the CLVS, the il_num_sets_in_message_minus1-th identified tile set consists of those remaining tiles that are not included in any earlier identified tile sets in the same message and all the prediction blocks that are inside the il_num_sets_in_message_minus1-th identified tile set are inter-layer predicted from inter-layer reference pictures with nuh_layer_id equal to IdDirectRefLayer[targetLayerId][NumDirectRefLayers[targetLayerId] − 1] and no residual_coding( ) syntax structure is present in any transform unit of the il_num_sets_in_message_minus1-th identified tile set. skipped_tile_set_present_flag equal to 0 does not indicate a bitstream constraint within the CLVS. When not present, the value of skipped_tile_set_present_flag is inferred to be equal to 0.

ilcts_id[i] contains an identifying number that may be used to identify the purpose of the i-th identified tile set (for example, to identify an area to be extracted from the coded video sequence for a particular purpose). The value of ilcts_id[i] shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of ilcts_id[i] from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. If il_one_tile_per_tile_set_flag is equal to 1, values of ilcts_id[i] from 256 to 511, inclusive, are used for the inferred ilcts_id[i] values as specified above. Otherwise, values of ilcts_id[i] from 256 to 511, inclusive, are reserved for future use by ITU-T | ISO/IEC. Values of ilcts_id[i] from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering an indicated value of ilcts_id[i] in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

When ilcts_id[i] is not present for i in the range of 0 to (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1, inclusive, due to il_one_tile_per_tile_set_flag being equal to 1, it is inferred to be equal to 256 + i.

il_num_tile_rects_in_set_minus1[i] plus 1 specifies the number of rectangular regions of tiles in the i-th identified tile set. The value of il_num_tile_rects_in_set_minus1[i] shall be in the range of 0 to (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1, inclusive.

il_top_left_tile_idx[i][j] and **il_bottom_right_tile_idx**[i][j] identify the tile position of the top-left tile and the tile position of the bottom-right tile in a rectangular region of the i-th identified tile set, respectively, in tile raster scan order.

ilc_idc[i] equal to 1 indicates that no samples outside of any associated reference tile set and no samples at a fractional sample position that is derived using one or more samples outside of any associated reference tile set are used for inter-layer prediction of any sample within the i-th identified tile set. ilc_idc[i] equal to 2 indicates that no prediction block in

the i-th identified tile set is predicted from an inter-layer reference picture. ilc_idc[i] equal to 0 indicates that the inter-layer prediction process may or may not be constrained for the prediction block in the i-th identified tile set. The value of ilc_idc[i] equal to 3 is reserved.

il_exact_sample_value_match_flag[i] equal to 1 indicates that, within the CLVS, when the CTBs that do not belong to the i-th identified tile set are not decoded and the boundaries of the i-th identified tile set are treated as picture boundaries for the purposes of the decoding process, the value of each sample in the i-th identified tile set would be exactly the same as the value of the sample that would be obtained when all the CTBs of all pictures in the CLVS are decoded. il_exact_sample_value_match_flag[i] equal to 0 indicates that, within the CLVS, when the CTBs that are outside of the i-th identified tile set are not decoded and the boundaries of the i-th identified tile set are treated as picture boundaries for the purposes of the decoding process, the value of each sample in the i-th identified tile set may or may not be exactly the same as the value of the same sample when all the CTBs of all pictures in the CLVS are decoded.

NOTE 2 – It is feasible to use il_exact_sample_value_match_flag equal to 1 when using certain combinations of loop_filter_across_tiles_enabled_flag, pps_loop_filter_across_slices_enabled_flag, pps_deblocking_filter_disabled_flag, slice_loop_filter_across_slices_enabled_flag, slice_deblocking_filter_disabled_flag, sample_adaptive_offset_enabled_flag, slice_sao_luma_flag and slice_sao_chroma_flag.

all_tiles_ilc_idc equal to 1 indicates that, for each identified tile set within the CLVS, no sample value outside of each associated reference tile set and no sample value at a fractional sample position that is derived using one or more samples outside of each associated reference tile set is used for inter-layer prediction of any sample within the identified tile set. all_tiles_ilc_idc equal to 2 indicates that, within the CLVS, no prediction block in each identified tile set is predicted from an inter-layer reference picture. all_tiles_ilc_idc equal to 0 indicates that, within the CLVS, the inter-layer prediction process may or may not be constrained for the identified tile sets. The value of all_tiles_ilc_idc equal to 3 is reserved. When all_tiles_ilc_idc is not present, it is inferred to be equal to 0.

F.14.3.5 Bitstream partition nesting SEI message semantics

The bitstream partition nesting SEI message provides a mechanism to associate SEI messages with a bitstream partition of a partitioning scheme of an OLS.

NOTE – The bitstream partition nesting SEI message must be contained within a scalable nesting SEI message. Constraints on the scalable nesting SEI message containing a bitstream partition nesting SEI message are specified in clause F.14.3.2.7.

A bitstream partition nesting SEI message contains one or more SEI messages.

sei_ols_idx specifies the index of the OLS to which the contained SEI messages apply. The value of sei_ols_idx shall be in the range of 0 to NumOutputLayerSets − 1, inclusive.

It is a requirement of bitstream conformance that OlsIdxToLsIdx[sei_ols_idx] shall be equal to nesting_op_idx[0] of the scalable nesting SEI message that contains the bitstream partition nesting SEI message.

sei_partitioning_scheme_idx specifies the index of the partitioning scheme to which the contained SEI messages apply. The value of sei_partitioning_scheme_idx shall be in the range of 0 to num_signalled_partitioning_schemes[sei_ols_idx], inclusive.

bsp_idx specifies the index of the bitstream partition to which the contained SEI messages apply. The value of bsp_idx shall be in the range of 0 to num_partitions_in_scheme_minus1[sei_ols_idx][sei_partitioning_scheme_idx], inclusive.

num_seis_in_bsp_minus1 plus 1 specifies the number of sei_message( ) syntax structures contained in the bsp_nesting( ) syntax structure. The value of num_seis_in_bsp_minus1 shall be in the range of 0 to 63, inclusive.

bsp_nesting_zero_bit shall be equal to 0.

F.14.3.6 Bitstream partition initial arrival time SEI message semantics

The bitstream partition initial arrival time SEI message specifies the initial arrival times to be used in the bitstream-partition-specific CPB operation.

When present, this SEI message shall be contained within a bitstream partition nesting SEI message that is contained in a scalable nesting SEI message, and the same bitstream partition nesting SEI message shall also contain a buffering period SEI message.

The following applies for each value of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] − 1, inclusive:

– The variable hrdParamIdx[i] is set equal to the value of bsp_hrd_idx[olsIdx][psIdx][maxTemporalId[0]][i][bspIdx], where olsIdx, psIdx and bspIdx are equal to sei_ols_idx, sei_partitioning_scheme_idx and bsp_idx, respectively, of the bitstream partition nesting SEI message containing this bitstream partition initial arrival time SEI message, and maxTemporalId[0] is the value of

MaxTemporalId[0] for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this bitstream partition initial arrival time SEI message.

– The variable initialCpbRemovalDelayLength[i] is set equal to initial_cpb_removal_delay_length_minus1 + 1, where initial_cpb_removal_delay_length_minus1 is found in the hrdParamIdx[i]-th hrd_parameters( ) syntax structure in the active VPS.

– The variable nalHrdParamsPresent[i] is set equal to the value of nal_hrd_parameters_present_flag in the hrdParamIdx[i]-th hrd_parameters( ) syntax structure in the active VPS.

– The variable vclHrdParamsPresent[i] is set equal to the value of vcl_hrd_parameters_present_flag in the hrdParamIdx[i]-th hrd_parameters( ) syntax structure in the active VPS.

It is a requirement of bitstream conformance that the value of nalHrdParamsPresent[i] shall be the same for all values of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] − 1, inclusive.

It is a requirement of bitstream conformance that the value of vclHrdParamsPresent[i] shall be the same for all values of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] − 1, inclusive.

The variable nalInitialArrivalDelayPresent is set equal to nalHrdParamsPresent[i] of any value of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] − 1, inclusive.

The variable vclInitialArrivalDelayPresent is set equal to vclHrdParamsPresent[i] of any value of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] − 1, inclusive.

nal_initial_arrival_delay[i] specifies the initial arrival time for the i-th delivery schedule of the bitstream partition to which this SEI message applies, when NAL HRD parameters are in use. The length, in bits, of the nal_initial_arrival_delay[i] syntax element is equal to initialCpbRemovalDelayLength[i].

vcl_initial_arrival_delay[i] specifies the initial arrival time for the i-th delivery schedule of the bitstream partition to which this SEI message applies, when VCL HRD parameters are in use. The length, in bits, of the vcl_initial_arrival_delay[i] syntax element is equal to initialCpbRemovalDelayLength[i].

F.14.3.7 Sub-bitstream property SEI message semantics

The sub-bitstream property SEI message, when present, provides the bit rate information for a sub-bitstream created by discarding those pictures in the layers that do not belong to the output layers of the OLSs specified by the active VPS and that do not affect the decoding of the output layers.

When present, the sub-bitstream property SEI message shall be associated with an initial IRAP access unit and the information provided by the SEI messages applies to the bitstream corresponding to the CVS containing the associated initial IRAP access unit.

sb_property_active_vps_id identifies the active VPS. The value of sb_property_active_vps_id shall be equal to the value of vps_video_parameter_set_id of the active VPS referred to by the VCL NAL units of the associated access unit.

num_additional_sub_streams_minus1 plus 1 specifies the number of the sub-bitstreams for which the bit rate information may be provided by this SEI message. The value of num_additional_sub_streams_minus1 shall be in the range of 0 to $2^{10} - 1$, inclusive.

sub_bitstream_mode[i] specifies how the i-th sub-bitstream is generated. The value of sub_bitstream_mode[i] shall be equal to 0 or 1, inclusive. The values 2 and 3 are reserved for future use by ITU-T and ISO/IEC. When sub_bitstream_mode[i] is the greater than 1, decoders shall ignore the syntax elements ols_idx_to_vps[i], highest_sublayer_id[i], avg_sb_property_bit_rate[i] and max_sb_property_bit_rate[i].

When sub_bitstream_mode[i] is equal to 0, the i-th sub-bitstream is generated as follows:

– Let lsIdx be equal to OlsIdxToLsIdx[ols_idx_to_vps[i]].

– A sub-bitstream subBitstream[i] is first created as follows:

  – If lsIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 1, the sub-bitstream extraction process as specified in clause 10 is invoked with the bitstream corresponding to the CVS containing the sub-bitstream property SEI message, highest_sublayer_id[i] and LayerSetLayerIdList[lsIdx] as inputs, and the output is assigned to subBitstream[i].

  – Otherwise, if lsIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 0, the sub-bitstream extraction process as specified in clause F.10.1 is invoked with the bitstream corresponding to the CVS containing the sub-bitstream property SEI message, highest_sublayer_id[i] and LayerSetLayerIdList[lsIdx] as inputs, and the output is assigned to subBitstream[i].

  - Otherwise, the sub-bitstream extraction process as specified in clause F.10.3 is invoked with the bitstream corresponding to the CVS containing the sub-bitstream property SEI message, highest_sublayer_id[i] and LayerSetLayerIdList[lsIdx] as inputs, and the output is assigned to subBitstream[i].

- Remove all NAL units for which the nuh_layer_id is not included in TargetOptLayerIdList and either of the following conditions is true:
  - The value of nal_unit_type is not in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, and max_tid_il_ref_pics_plus1[LayerIdxInVps[nuh_layer_id]][LayerIdxInVps[layerId]] is equal to 0 for layerId values included in TargetOptLayerIdList.
  - TemporalId is greater than the maximum value of max_tid_il_ref_pics_plus1[LayerIdxInVps[nuh_layer_id]][LayerIdxInVps[layerId]] − 1 for all layerId values included in TargetOptLayerIdList.

When sub_bitstream_mode[i] is equal to 1, the i-th sub-bitstream is generated as specified by the above process followed by:

- Remove all NAL units with nuh_layer_id not among the values included in TargetOptLayerIdList and with discardable_flag equal to 1.

ols_idx_to_vps[i] specifies the index of the OLS corresponding to the i-th sub-bitstream. The value of ols_idx_to_vps[i] shall be in the range of 0 to NumOutputLayerSets − 1, inclusive.

highest_sublayer_id[i] specifies the highest TemporalId of access units in the i-th sub-bitstream.

avg_sb_property_bit_rate[i] indicates the average bit rate of the i-th sub-bitstream, in bits per second. The value is given by BitRateBPS(avg_sb_property_bit_rate[i]) with the function BitRateBPS( ) being specified as follows:

$$\text{BitRateBPS}(\ x\ ) = (\ x\ \&\ (\ 2^{14} - 1\ )\ )\ *\ 10^{(\ 2\ +\ (\ x\ >>\ 14\ )\ )} \quad \text{(F-91)}$$

The average bit rate is derived according to the access unit removal time specified in clause F.13. In the following, bTotal is the number of bits in all NAL units of the i-th sub-bitstream, t_1 is the removal time (in seconds) of the first access unit to which the VPS applies and t_2 is the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies. With x specifying the value of avg_sb_property_bit_rate[i], the following applies:

- If t_1 is not equal to t_2, the following condition shall be true:

$$(\ x\ \&\ (\ 2^{14} - 1\ )\ )\ ==\ \text{Round}(\ \text{bTotal} \div (\ (\ t_2 - t_1\ )\ *\ 10^{(\ 2\ +\ (\ x\ >>\ 14\ )\ )}\ )\ ) \quad \text{(F-92)}$$

- Otherwise (t_1 is equal to t_2), the following condition shall be true:

$$(\ x\ \&\ (\ 2^{14} - 1\ )\ )\ ==\ 0 \quad \text{(F-93)}$$

max_sb_property_bit_rate[i] indicates an upper bound for the bit rate of the i-th sub-bitstream in any one-second time window of access unit removal time as specified in clause F.13. The upper bound for the bit rate in bits per second is given by BitRateBPS(max_sb_property_bit_rate[i]). The bit rate values are derived according to the access unit removal time specified in clause F.13. In the following, t_1 is any point in time (in seconds), t_2 is set equal to $t_1 + 1 \div 100$ and bTotal is the number of bits in all NAL units of access units with a removal time greater than or equal to t_1 and less than t_2. With x specifying the value of max_sb_property_bit_rate[i], the following condition shall be obeyed for all values of t_1:

$$(\ x\ \&\ (\ 2^{14} - 1\ )\ )\ >=\ \text{bTotal} \div (\ (\ t_2 - t_1\ )\ *\ 10^{(\ 2\ +\ (\ x\ >>\ 14\ )\ )}\ ) \quad \text{(F-94)}$$

F.14.3.8 Alpha channel information SEI message semantics

The alpha channel information SEI message provides information about alpha channel sample values and post-processing applied to the decoded alpha planes coded in auxiliary pictures of type AUX_ALPHA and one or more associated primary pictures.

For an auxiliary picture with nuh_layer_id equal to nuhLayerIdA and AuxId[nuhLayerIdA] equal to AUX_ALPHA, an associated primary picture, if any, is a picture in the same access unit having AuxId[nuhLayerIdB] equal to 0 such that ScalabilityId[LayerIdxInVps[nuhLayerIdA]][j] is equal to ScalabilityId[LayerIdxInVps[nuhLayerIdB]][j] for all values of j in the range of 0 to 2, inclusive, and 4 to 15, inclusive.

When an access unit contains an auxiliary picture picA with nuh_layer_id equal to nuhLayerIdA and AuxId[nuhLayerIdA] equal to AUX_ALPHA, the alpha channel sample values of picA persist in output order until one or more of the following conditions are true:

- The next picture, in output order, with nuh_layer_id equal to nuhLayerIdA is output.
- A CLVS containing the auxiliary picture picA ends.
- The bitstream ends.

– A CLVS of any associated primary layer of the auxiliary picture layer with nuh_layer_id equal to nuhLayerIdA ends.

The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the alpha channel information SEI message applies.

alpha_channel_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous alpha channel information SEI message in output order that applies to the current layer. alpha_channel_cancel_flag equal to 0 indicates that alpha channel information follows.

Let currPic be the picture that the alpha channel information SEI message is associated with. The semantics of alpha channel information SEI message persist for the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture picB with nuh_layer_id equal to targetLayerId in an access unit containing an alpha channel information SEI message with nuh_layer_id equal to targetLayerId is output having PicOrderCnt(picB) greater than PicOrderCnt(currPic), where PicOrderCnt(picB) and PicOrderCnt(currPic) are the PicOrderCntVal values of picB and currPic, respectively, immediately after the invocation of the decoding process for picture order count for picB.

alpha_channel_use_idc equal to 0 indicates that for alpha blending purposes the decoded samples of the associated primary picture should be multiplied by the interpretation sample values of the auxiliary coded picture in the display process after output from the decoding process. alpha_channel_use_idc equal to 1 indicates that for alpha blending purposes the decoded samples of the associated primary picture should not be multiplied by the interpretation sample values of the auxiliary coded picture in the display process after output from the decoding process. alpha_channel_use_idc equal to 2 indicates that the usage of the auxiliary picture is unspecified. Values greater than 2 for alpha_channel_use_idc are reserved for future use by ITU-T | ISO/IEC. When not present, the value of alpha_channel_use_idc is inferred to be equal to 2.

alpha_channel_bit_depth_minus8 plus 8 specifies the bit depth of the samples of the luma sample array of the auxiliary picture. alpha_channel_bit_depth_minus8 shall be in the range 0 to 7 inclusive. alpha_channel_bit_depth_minus8 shall be equal to bit_depth_luma_minus8 of the associated primary picture.

alpha_transparent_value specifies the interpretation sample value of an auxiliary coded picture luma sample for which the associated luma and chroma samples of the primary coded picture are considered transparent for purposes of alpha blending. The number of bits used for the representation of the alpha_transparent_value syntax element is alpha_channel_bit_depth_minus8 + 9.

alpha_opaque_value specifies the interpretation sample value of an auxiliary coded picture luma sample for which the associated luma and chroma samples of the primary coded picture are considered opaque for purposes of alpha blending. The number of bits used for the representation of the alpha_opaque_value syntax element is alpha_channel_bit_depth_minus8 + 9.

alpha_channel_incr_flag equal to 0 indicates that the interpretation sample value for each decoded auxiliary picture luma sample value is equal to the decoded auxiliary picture sample value for purposes of alpha blending. alpha_channel_incr_flag equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample value that is greater than Min(alpha_opaque_value, alpha_transparent_value) should be increased by one to obtain the interpretation sample value for the auxiliary picture sample and any auxiliary picture luma sample value that is less than or equal to Min(alpha_opaque_value, alpha_transparent_value) should be used, without alteration, as the interpretation sample value for the decoded auxiliary picture sample value. When not present, the value of alpha_channel_incr_flag is inferred to be equal to 0.

alpha_channel_clip_flag equal to 0 indicates that no clipping operation is applied to obtain the interpretation sample values of the decoded auxiliary picture. alpha_channel_clip_flag equal to 1 indicates that the interpretation sample values of the decoded auxiliary picture are altered according to the clipping process described by the alpha_channel_clip_type_flag syntax element. When not present, the value of alpha_channel_clip_flag is inferred to be equal to 0.

alpha_channel_clip_type_flag equal to 0 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample that is greater than (alpha_opaque_value − alpha_transparent_value) / 2 is set equal to alpha_opaque_value to obtain the interpretation sample value for the auxiliary picture luma sample and any auxiliary picture luma sample that is less or equal than (alpha_opaque_value − alpha_transparent_value) / 2 is set equal to alpha_transparent_value to obtain the interpretation sample value for the auxiliary picture luma sample. alpha_channel_clip_type_flag equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample that is greater than alpha_opaque_value is set equal to alpha_opaque_value to obtain the interpretation sample value for the auxiliary picture luma sample and any auxiliary picture luma sample that is less than or equal to alpha_transparent_value is set equal to

alpha_transparent_value to obtain the interpretation sample value for the auxiliary picture luma sample.

NOTE – When both alpha_channel_incr_flag and alpha_channel_clip_flag are equal to one, the clipping operation specified by alpha_channel_clip_type_flag should be applied first followed by the alteration specified by alpha_channel_incr_flag to obtain the interpretation sample value for the auxiliary picture luma sample.

F.14.3.9 Overlay information SEI message semantics

The overlay information SEI message provides information about overlay pictures coded as auxiliary pictures. Overlay auxiliary pictures have nuh_layer_id equal to nuhLayerIdA and AuxId[nuhLayerIdA] in the range of 128 to 159, inclusive. Each overlay auxiliary picture layer is associated with one or more primary picture layers as specified below.

overlay_info_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous overlay information SEI message in output order that is associated with one or more primary picture layers to which this SEI applies. overlay_info_cancel_flag equal to 0 indicates that overlay information follows.

overlay_content_aux_id_minus128 plus 128 indicates the value of AuxId of auxiliary pictures containing overlay content. overlay_content_aux_id_minus128 shall be in the range of 0 to 31, inclusive.

overlay_label_aux_id_minus128 plus 128 indicates the value of AuxId of auxiliary pictures containing overlay label. overlay_label_aux_id_minus128 shall be in the range of 0 to 31, inclusive.

overlay_alpha_aux_id_minus128 plus 128 indicates the value of AuxId of auxiliary pictures containing overlay alpha. overlay_alpha_aux_id_minus128 shall be in the range of 0 to 31, inclusive.

overlay_element_label_value_length_minus8 plus 8 indicates the number of bits used for coding the overlay_element_label_min[i][j] and overlay_element_label_max[i][j] syntax elements.

num_overlays_minus1 plus 1 specifies the number of overlays described by the overlay information SEI message. num_overlays_minus1 shall be in the range of 0 to 15, inclusive.

overlay_idx[i] indicates the index of the i-th overlay. overlay_idx[i] shall be in the range of 0 to 255, inclusive.

language_overlay_present_flag[i] equal to 1 indicates that overlay_language[i] is present. language_overlay_present_flag[i] equal to 0 indicates that overlay_language[i] is not present and that the language of the overlay is unspecified.

overlay_content_layer_id[i] indicates the nuh_layer_id value of the NAL units of the overlay content of the i-th overlay. AuxId[overlay_content_layer_id[i]] shall be equal to overlay_content_aux_id_minus128 + 128 for all values of i in the range of 0 to num_overlays_minus1, inclusive.

The value of the variable pLid, which identifies the nuh_layer_id value of the primary picture which the i-th overlay is associated with, is derived as follows:

```
pLid = −1
for( j = 0; j < 63; j++ )
    if( ViewOrderIdx[ j ] = = ViewOrderIdx[ overlay_content_layer_id[ i ] ] &&              (F-95)
        DependencyId[ j ] = = DependencyId[ overlay_content_layer_id[ i ] ] && AuxId[ j ] = =
0 )
        pLid = j
```

The value of pLid shall be in the range of 0 to 62, inclusive.

overlay_label_present_flag[i] equal to 1 specifies that overlay_label_layer_id[i] is present. overlay_label_present_flag[i] equal to 0 specifies that overlay_label_layer_id[i] is not present.

overlay_label_layer_id[i] indicates the nuh_layer_id value of NAL units in the overlay label of the i-th overlay. AuxId[overlay_label_layer_id[i]] shall be equal to overlay_label_aux_id_minus128 + 128 for all values of i in the range of 0 to num_overlays_minus1, inclusive.

overlay_alpha_present_flag[i] equal to 1 specifies that overlay_alpha_layer_id[i] is present. overlay_alpha_present_flag[i] equal to 0 specifies that overlay_alpha_layer_id[i] is not present.

overlay_alpha_layer_id[i] indicates the nuh_layer_id value of NAL units in the overlay alpha of the i-th overlay. AuxId[overlay_alpha_layer_id[i]] shall be equal to overlay_alpha_aux_id_minus128 + 128 for all values of i in the range of 0 to num_overlays_minus1, inclusive.

num_overlay_elements_minus1[i] indicates the number of overlay elements in the i-th overlay. num_overlay_elements_minus1[i] shall be in the range of 0 to 255, inclusive. When not present, the value of num_overlay_elements_minus1[i] is inferred to be equal to 0.

overlay_element_label_min[i][j] and **overlay_element_label_max**[i][j] indicate the minimum and maximum values, respectively, of the range of sample values corresponding to the j-th overlay element of the i-th overlay. The length of the overlay_element_label_min[i][j] and the overlay_element_label_max[i][j] syntax elements is overlay_element_label_min_max_length_minus8 + 8 bits.

The variable overlayElementId[i][x][y] specifying the overlay element identifier of the i-th overlay for the sample location (x, y) relative to the top-left sample is derived as follows, where p[i][x][y] refers to the luma sample at location (x, y) in the decoded label auxiliary picture of the i-th overlay:

```
for( y = 0; y < pic_height_in_luma_samples; y++ )
    for( x = 0; x < pic_width_in_luma_samples; x++ )
        for( i = 0; i <= number_overlays_minus1[ i ] ) {
            overlayElementId[ i ][ x ][ y ] = 0
            for( j = 0; j <= num_overlay_elements_minus1[ i ]; j++ )            (F-96)
                if( p[ i ][ x ][ y ] >= overlay_element_label_min[ i ][ j ] &&
                    p[ i ][ x ][ y ] <= overlay_element_label_max[ i ][ j ] )
                    overlayElementId[ i ][ x ][ y ] = j
        }
```

overlay_zero_bit shall be equal to 0.

overlay_language[i] contains a language tag as specified by IETF RFC 5646 followed by a null termination byte equal to 0x00. The length of the overlay_language[i] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

overlay_name[i] indicates the name of the i-th overlay. The length of the overlay_name[i] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

overlay_element_name[i][j] indicates the name of the j-th overlay element of the i-th overlay. The length of the overlay_element_name[i][j] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

overlay_info_persistence_flag specifies the persistence of the overlay information SEI message. overlay_info_persistence_flag equal to 0 specifies that the overlay information SEI message applies to the current decoded picture only.

When an access unit contains an auxiliary picture picA with nuh_layer_id equal to nuhLayerIdA and AuxId[nuhLayerIdA] in the range of 128..159, and nuhLayerIdA is equal to any of overlay_content_layer_id[i], overlay_label_layer_id[i], overlay_label_layer_id[i] and overlay_alpha_layer_id[i], overlay_info_persistence_flag equal to 1 specifies that the overlay information SEI message persists for the CLVS containing the auxiliary picture picA in output order until one or more of the following conditions are true:

– A new CLVS begins for the layer containing the auxiliary picture picA.
– A new CLVS begins for the layer containing the primary picture associated with the auxiliary picture picA.
– The bitstream ends.
– A picture picB in an access unit containing an overlay information SEI message is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

F.14.3.10 Temporal motion vector prediction constraints SEI message semantics

The temporal motion vector prediction constraints SEI message indicates constraints on collocated pictures for temporal motion vector prediction. This SEI message may be used to determine whether the motion vectors of earlier pictures in decoding order no longer need to be stored and whether the motion vectors of the current picture and subsequent pictures need to be stored.

The temporal motion vector prediction constraints SEI message is a prefix SEI message. The temporal motion vector prediction constraints SEI message may be present in an access unit with TemporalId equal to 0 and shall not be present in an access unit with TemporalId greater than 0.

The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the temporal motion vector prediction constraints SEI message applies. Let associatedLayerIdList consist of each targetLayerId value to which this temporal motion vector prediction constraints SEI message applies.

Let a set of pictures associatedPicSet be the pictures with nuh_layer_id equal to targetLayerId from the access unit containing the SEI message, inclusive, up to but not including the first of any of the following in decoding order:

– The next access unit, in decoding order, that contains a temporal motion vector prediction constraints SEI message with an associatedLayerIdList that contains targetLayerId.

– The first picture of the next CLVS, in decoding order, of the layer with nuh_layer_id equal to targetLayerId.

prev_pics_not_used_flag equal to 1 indicates that the syntax elements for all coded pictures that are within or follow the access unit containing the current picture in decoding order are constrained such that no temporal motion vector from any picture that has nuh_layer_id equal to any value in associatedLayerIdList and precedes the access unit containing the current picture in decoding order is used directly or indirectly in decoding of any coded picture that is within or follows the access unit containing the current picture in decoding order. prev_pics_not_used_flag equal to 0 indicates that the bitstream may or may not fulfill the constraints indicated by prev_pics_not_used_flag equal to 1.

NOTE 1 – When prev_pics_not_used_flag is equal to 1, decoders may empty the "motion vector storage" for all reference pictures with nuh_layer_id equal to targetLayerId in the decoded picture buffer.

prev_pics_not_used_flag shall be equal to 1 when both of the following conditions are true:

– no_intra_layer_col_pic_flag is equal to 1 in the previous temporal motion vector prediction constraints SEI message applying to nuh_layer_id equal to targetLayerId.

– The previous temporal motion vector prediction constraints SEI message applying to nuh_layer_id equal to targetLayerId and the current temporal motion vector prediction constraints SEI message apply to the same CLVS of the layer with nuh_layer_id equal to targetLayerId.

no_intra_layer_col_pic_flag equal to 1 indicates the following:

– If NumDirectRefLayers[targetLayerId] is equal to 0, slice_temporal_mvp_enabled_flag is not present or is equal to 0 in each picture in associatedPicSet.

– Otherwise, all the pictures in associatedPicSet do not use temporal motion vector prediction or use collocated pictures with nuh_layer_id different from targetLayerId.

When no_intra_layer_col_pic_flag is equal to 0, no constraint on the collocated picture of the pictures with nuh_layer_id equal to targetLayerId is indicated.

Let NoIntraLayerColPicFlag[targetLayerId] be equal to no_intra_layer_col_pic_flag.

NOTE 2 – The motion vectors of the current picture with nuh_layer_id equal to layerId have to be stored when they may be used for temporal motion vector prediction of other pictures in the same layer or when they may be used for inter-layer motion prediction. In other words, the motion vectors of the current picture have to be stored when at least one of the following is true:

- sps_temporal_mvp_enabled_flag in the active SPS for the current picture is equal to 1 and NoIntraLayerColPicFlag[layerId] is equal 0.
- NoIntraLayerColPicFlag[layerId] is equal to 1 and there is a nuh_layer_id value nuhLayerIdA such that VpsInterLayerMotionPredictionEnabled[LayerIdxInVps[nuhLayerIdA]][LayerIdxInVps[layerId]] is equal to 1.

NOTE 3 – The motion vectors of a picture with nuh_layer_id equal to layerId need no longer be stored when the picture is marked as "unused for reference", or the picture is not used for temporal motion vector prediction of other pictures in the same layer and all pictures in the same access unit that may use the picture as a reference for inter-layer motion prediction have been decoded, or the access unit containing the picture precedes the current access unit in decoding order, where this SEI message is present with associatedLayerIdList including the nuh_layer_id of the picture and prev_pics_not_used_flag equal to 1.

F.14.3.11 Frame-field information SEI message semantics

The frame-field information SEI message may be used to indicate how the associated picture should be displayed, the source scan type of the associated picture, and whether the associated picture is a duplicate of a previous picture, in output order, of the same layer.

The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the frame-field information SEI message applies.

A frame-field information SEI message associated with nuh_layer_id equal to targetLayerId shall be present in an access unit, when all of the following conditions are true:

– A picture with nuh_layer_id equal to targetLayerId is present in the access unit.

– frame_field_info_present_flag is equal to 1 in the active SPS for the layer with nuh_layer_id equal to targetLayerId.

– targetLayerId is greater than 0 or none of the following conditions is true:

  – A non-scalable-nested picture timing SEI message is present in the access unit.

  – A picture timing SEI message directly contained in a scalable nesting SEI message is present in the access unit.

A frame-field information SEI message that applies to a particular set of layers shall not be present when one or more of the following conditions are true:

– The value of field_seq_flag is not the same for all active SPSs for the particular set of layers.

– The values of general_progressive_source_flag and general_interlaced_source_flag are not identical, respectively, for all the profile_tier_level( ) syntax structures that apply to the layers to which the frame-field information SEI message applies.

The semantics of ffinfo_pic_struct, ffinfo_source_scan_type and ffinfo_duplicate_flag apply layer-wise to each value of targetLayerId.

ffinfo_pic_struct has the same semantics as the pic_struct syntax element in the picture timing SEI message.

ffinfo_source_scan_type has the same semantics as the source_scan_type syntax element in the picture timing SEI message.

ffinfo_duplicate_flag has the same semantics as the duplicate_flag syntax element in the picture timing SEI message.

F.15 Video usability information

F.15.1 General

The specifications in clause E.1 apply.

F.15.2 VUI syntax

The specifications in clause E.2 and its subclauses apply.

F.15.3 VUI semantics

F.15.3.1 VUI parameters semantics

The specifications in clause E.3.1 apply by replacing the semantics of field_seg_flag, frame_field_info_present_flag and vui_timing_info_present_flag with those below and with the following additions:

– It is a requirement of bitstream conformance that, when nuh_layer_id layerId is greater than 0 and NumDirectRefLayers[layerId] is greater than 0, video_signal_type_present_flag present in or inferred for the active SPS for nuh_layer_id equal to layerId shall be equal to 0.

– When the current picture has nuh_layer_id layerIdCurr greater than 0, either NumDirectRefLayers[layerIdCurr] is greater than 0 or MultiLayerExtSpsFlag derived from the active SPS for the nuh_layer_id equal to layerIdCurr is equal to 1 and the active SPS for nuh_layer_id equal to layerIdCurr contains the VUI parameters syntax structure, the following applies:

  – The values of video_format, video_full_range_flag, colour_primaries, transfer_characteristics and matrix_coeffs are inferred to be equal to video_vps_format, video_full_range_vps_flag, colour_primaries_vps, transfer_characteristics_vps and matrix_coeffs_vps, respectively, of the vps_video_signal_info_idx[j]-th video_signal_info( ) syntax structure in the active VPS where j is equal to LayerIdxInVps[layerIdCurr].

  – The values of video_format, video_full_range_flag, colour_primaries, transfer_characteristics and matrix_coeffs signalled in the active SPS for the layer with nuh_layer_id equal to layerIdCurr are ignored.

NOTE 1 – The values are inferred from the VPS when a non-base layer refers to an SPS that is also referred to by the base layer, in which case the SPS has nuh_layer_id equal to 0. For the base layer, the values of these parameters in the active SPS for the base layer apply.

field_seq_flag equal to 1 indicates that the layers for which the SPS is an active SPS within the CVS convey pictures that represent fields and specifies the following:

– When the SPS is an active SPS for nuh_layer_id equal to 0, a picture timing SEI message that is not scalable-nested or that is directly contained in a scalable nesting SEI message and applies to nuh_layer_id equal to 0 shall be present in every such access unit of the current CVS that contains a coded picture with nuh_layer_id equal to 0.

– When the SPS is an active SPS for the nuh_layer_id value nuhLayerId greater than 0, a frame-field information SEI message shall be present for nuh_layer_id equal to nuhLayerId in every access unit containing a picture for the current CLVS of the layer with nuh_layer_id equal to nuhLayerId.

field_seq_flag equal to 0 indicates that the layers for which the SPS is an active SPS within the CVS convey pictures that represent frames and that a picture timing SEI message or a frame-field information SEI message may or may not be present for the pictures within the CVS belonging to any layer for which the SPS is an active SPS. When field_seq_flag is not present, it is inferred to be equal to 0. When general_frame_only_constraint_flag is present in the SPS and is equal to 1, the value of field_seq_flag shall be equal to 0. When general_frame_only_constraint_flag is present in the active VPS, applies for a layer for which the SPS is an active SPS and is equal to 1, the value of field_seq_flag shall be equal to 0.

NOTE 2 – The specified decoding process does not treat access units conveying pictures that represent fields or frames differently. A sequence of pictures that represent fields would therefore be coded with the picture dimensions of an individual field. For example, access units containing pictures that represent 1080i fields would commonly have cropped output dimensions of 1920x540, while the sequence picture rate would commonly express the rate of the source fields (typically between 50 and 60 Hz), instead of the source frame rate (typically between 25 and 30 Hz).

frame_field_info_present_flag equal to 1 specifies that picture timing SEI messages or frame-field information SEI messages are present for every picture for which this SPS is the active SPS and the picture timing SEI messages, when present, include the pic_struct, source_scan_type, and duplicate_flag syntax elements. frame_field_info_present_flag equal to 0 specifies that the pic_struct syntax element is not present in picture timing SEI messages associated with pictures for which the SPS is the active SPS.

When frame_field_info_present_flag is present and either or both of the following conditions are true, frame_field_info_present_flag shall be equal to 1:

– field_seq_flag is equal to 1.

– general_progressive_source_flag and general_interlaced_source_flag are present in this SPS, general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 1.

When frame_field_info_present_flag is not present, its value is inferred as follows:

– If general_progressive_source_flag and general_interlaced_source_flag are present in this SPS, general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 1, frame_field_info_present_flag is inferred to be equal to 1.

– Otherwise, frame_field_info_present_flag is inferred to be equal to 0.

vui_timing_info_present_flag equal to 1 specifies that vui_num_units_in_tick, vui_time_scale, vui_poc_proportional_to_timing_flag and vui_hrd_parameters_present_flag are present in the vui_parameters( ) syntax structure. vui_timing_info_present_flag equal to 0 specifies that vui_num_units_in_tick, vui_time_scale, vui_poc_proportional_to_timing_flag and vui_hrd_parameters_present_flag are not present in the vui_parameters( ) syntax structure. It is a requirement of bitstream conformance that, when MultiLayerExtSpsFlag is equal to 1, vui_timing_info_present_flag shall be equal to 0.

F.15.3.2 HRD parameters semantics

The specifications in clause E.3.2 apply with the replacement of each instance of "access unit level" and "sub-picture level" with "partition unit level" and "sub-partition level", respectively, and with the following additions:

initial_cpb_removal_delay_length_minus1 plus 1 within the j-th hrd_parameters( ) syntax structure in the VPS specifies the length, in bits, of the nal_initial_arrival_delay[k] and vcl_initial_arrival_delay[k] syntax elements of the bitstream partition initial arrival time SEI message that is contained in a bitstream partition nesting SEI message within a scalable nesting SEI message with values of MaxTemporalId[0], sei_ols_idx, sei_partitioning_scheme_idx and bsp_idx such that bsp_hrd_idx[sei_ols_idx][sei_partitioning_scheme_idx][MaxTemporalId[0]][k][bsp_idx] is equal to j. When the initial_cpb_removal_delay_length_minus1 syntax element is not present, it is inferred to be equal to 23.

It is a requirement of bitstream conformance that the value of sub_pic_hrd_params_present_flag shall be the same for all hrd_parameters( ) syntax structures that apply to at least one bitstream partition of a particular output layer set.

F.15.3.3 Sub-layer HRD parameters semantics

The specifications in clause E.3.3 apply.

Annex G

Multiview high efficiency video coding

(This annex forms an integral part of this Recommendation | International Standard.)

G.1 Scope

This annex specifies syntax, semantics and decoding processes for multiview high efficiency video coding that use the syntax, semantics and decoding processes specified in clauses 2-10 and Annexes A-F. This annex also specifies profiles, tiers and levels for multiview high efficiency video coding.

G.2 Normative references

The list of normative references in clause F.2 applies.

G.3 Definitions

The specifications in clause F.3 and its subclauses apply.

G.4 Abbreviations

The specifications in clause F.4 apply.

G.5 Conventions

The specifications in clause F.5 apply.

G.6 Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships

The specifications in clause F.6 apply.

G.7 Syntax and semantics

The specifications in clause F.7 and its subclauses apply.

G.8 Decoding processes

G.8.1 General decoding process

G.8.1.1 General

The specifications of clause F.8.1.1 apply.

G.8.1.2 Decoding process for a coded picture with nuh_layer_id greater than 0

The decoding process for the current picture CurrPic is as follows:

1. The decoding of NAL units is specified in clause G.8.2.
2. The processes in clauses G.8.1.3 and F.8.3.4 specify the following decoding processes using syntax elements in the slice segment layer and above:
   - Prior to decoding the first slice of the current picture, clause G.8.1.3 is invoked.
   - At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause F.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
3. The processes in clauses G.8.4, G.8.5, G.8.6 and G.8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments and the division of the slice segments into CTUs each form a partitioning of the picture.

G.8.1.3 Decoding process for inter-layer reference picture set

Outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1.

The variable currLayerId is set equal to nuh_layer_id of the current picture.

The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 and the following applies:

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
    refPicSet0Flag =
        ( ( ViewId[ currLayerId ]  <=  ViewId[ 0 ]  &&  ViewId[ currLayerId ]  <=
ViewId[ RefPicLayerId[ i ] ] )  | |
        ( ViewId[ currLayerId ]  >=  ViewId[ 0 ]  &&  ViewId[ currLayerId ]  >=
ViewId[ RefPicLayerId[ i ] ] ) )
    if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
            nuh_layer_id equal to RefPicLayerId[ i ] ) {
        if( refPicSet0Flag ) {
            RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = picX
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term
reference"
        } else {
            RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = picX
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term
reference"
        }
    } else {
        if( refPicSet0Flag )
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"          (G-1)
        else
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] = "no reference picture"
    }
}
```

There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1.

There shall be no picture that has discardable_flag equal to 1 in RefPicSetInterLayer0 or RefPicSetInterLayer1.

If the current picture is a RADL picture, there shall be no entry in RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture.

NOTE – An access unit may contain both RASL and RADL pictures.

G.8.2 NAL unit decoding process

The specifications in clause F.8.2 apply.

G.8.3 Slice decoding processes

The specifications in clause F.8.3 and its subclauses apply.

G.8.4 Decoding process for coding units coded in intra prediction mode

The specifications in clause F.8.4 apply.

G.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in clause F.8.5 apply.

G.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause F.8.6 apply.

G.8.7 In-loop filter process

The specifications in clause F.8.7 apply.

G.9 Parsing process

The specifications in clause F.9 apply.

G.10 Specification of bitstream subsets

The specifications in clause F.10 and its subclauses apply.

G.11 Profiles, tiers and levels

G.11.1 Profiles

G.11.1.1 Multiview Main profile

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the Multiview Main profile, the following applies:

– Let olsIdx be the OLS index of the OLS, the sub-bitstream subBitstream and the base layer sub-bitstream baseBitstream are derived as specified in clause F.11.3.

When vps_base_layer_internal_flag is equal to 1, the base layer sub-bitstream baseBitstream shall obey the following constraints:

– The base layer sub-bitstream baseBitstream shall be indicated to conform to the Main profile.

The sub-bitstream subBitstream shall obey the following constraints:

– All active VPSs shall have vps_num_rep_formats_minus1 in the range of 0 to 15, inclusive.

– All active SPSs for layers in subBitstream shall have chroma_format_idc equal to 1 only.

– All active SPSs for layers in subBitstream shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag, extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag and cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.

– CtbLog2SizeY derived from all active SPSs for layers in subBitstream shall be in the range of 4 to 6, inclusive.

– All active PPSs for layers in subBitstream shall have log2_max_transform_skip_block_size_minus2 and chroma_qp_offset_list_enabled_flag, when present, equal to 0 only.

– ScalabilityId[j][smIdx] derived according to any active VPS shall be equal to 0 for any smIdx value not equal to 1 or 3 and for any value of j such that layer_id_in_nuh[j] is among layerIdListTarget that was used to derive subBitstream.

– When NumLayersInIdList[OlsIdxToLsIdx[olsIdx]] is equal to 2, output_layer_flag[olsIdx][j] derived according to any active VPS shall be equal to 1 for j in the range of 0 to 1, inclusive, for subBitstream.

– All active VPSs shall have alt_output_layer_flag[olsIdx] equal to 0 only.

– When ViewOrderIdx[i] derived according to any active VPS is equal to 1 for the layer with nuh_layer_id equal to i in subBitstream, inter_view_mv_vert_constraint_flag shall be equal to 1 in the sps_multilayer_extension( ) syntax structure in each active SPS for that layer.

– When ViewOrderIdx[i] derived according to any active VPS is greater than to 0 for the layer with nuh_layer_id equal to i in subBitstream, num_ref_loc_offsets shall be equal to 0 in each active PPS for that layer.

– When ViewOrderIdx[i] derived according to any active VPS is greater than 0 for the layer with nuh_layer_id equal to i in subBitstream, the values of pic_width_in_luma_samples and pic_height_in_luma_samples in each active SPS for that layer shall be equal to the values of pic_width_in_luma_samples and pic_height_in_luma_samples, respectively, in each active SPS for all reference layers of that layer.

– For a layer with nuh_layer_id iNuhLId equal to any value included in layerIdListTarget that was used to derive subBitstream, the value of NumRefLayers[iNuhLId], which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 4.

– All active SPSs for layers in subBitstream shall have sps_range_extension_flag and sps_scc_extension_flag equal to 0 only.

– All active PPSs for layers in subBitstream shall have pps_range_extension_flag and pps_scc_extension_flag equal to 0 only.

– All active SPSs for layers in subBitstream shall have bit_depth_luma_minus8 equal to 0 only.

– All active SPSs for layers in subBitstream shall have bit_depth_chroma_minus8 equal to 0 only.

– All active PPSs for layers in subBitstream shall have colour_mapping_enabled_flag equal to 0 only.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for any layer in subBitstream shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the Multiview Main profile in clause G.11.2 shall be fulfilled.

– For any active VPS, ViewOrderIdx[i] shall be greater than ViewOrderIdx[j] for any values of i and j among layerIdListTarget that was used to derive subBitstream such that AuxId[i] is equal to AuxId[j] and i is greater than j.

In the remainder of this clause and clause G.11.2.1, all syntax elements in the profile_tier_level( ) syntax structure refer to those in the profile_tier_level( ) syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the Multiview Main profile is indicated as follows:

– If OpTid of the output operation point is equal to vps_max_sub_layer_minus1, the conformance is indicated by general_profile_idc being equal to 6 or general_profile_compatibility_flag[6] being equal to 1 and general_max_12bit_constraint_flag being equal to 1, general_max_10bit_constraint_flag being equal to 1, general_max_8bit_constraint_flag being equal to 1, general_max_422chroma_constraint_flag being equal to 1, general_max_420chroma_constraint_flag being equal to 1, general_max_monochrome_constraint_flag being equal to 0, general_intra_constraint_flag being equal to 0 and general_one_picture_only_constraint_flag being equal to 0 and general_lower_bit_rate_constraint_flag being equal to 1.

– Otherwise (OpTid of the output operation point is less than vps_max_sub_layer_minus1), the conformance is indicated by sub_layer_profile_idc[OpTid] being equal to 6 or sub_layer_profile_compatibility_flag[OpTid][6] being equal to 1 and sub_layer_max_12bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_10bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_8bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_422chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_420chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_monochrome_constraint_flag[OpTid] being equal to 0, sub_layer_intra_constraint_flag[OpTid] being equal to 0 and sub_layer_one_picture_only_constraint_flag[OpTid] being equal to 0 and sub_layer_lower_bit_rate_constraint_flag[OpTid] being equal to 1.

G.11.2 Tiers and levels

G.11.2.1 General tier and level limits

For purposes of comparison of tier capabilities, the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 0 is considered to be a lower tier than the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 1.

For purposes of comparison of level capabilities, a particular level of a specific tier is considered to be a lower level than some other level of the same tier when the value of the general_level_idc or sub_layer_level_idc[i] of the particular level is less than that of the other level.

The following is specified for expressing the constraints in this clause and clause G.11.2.2:

– The value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor and MinCrScaleFactor is the same as that specified in Table A.10 for the Main profile.

– Let access unit n be the n-th access unit in decoding order, with the first access unit being access unit 0 (i.e., the 0-th access unit).

– Let the variable fR be set equal to 1 ÷ 300.

– Let the variable olsIdx be the index of the OLS.

– For each layer with nuh_layer_id equal to currLayerId, let the variable layerSizeInSamplesY be derived as follows:

layerSizeInSamplesY = pic_width_vps_in_luma_samples * pic_height_vps_in_luma_samples (G-2)

where pic_width_vps_in_luma_samples and pic_height_vps_in_luma_samples are found in the vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format( ) syntax structure in the VPS.

Each layer with nuh_layer_id equal to currLayerId conforming to a profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer and the CPB is understood to be the BPB:

a) The value of layerSizeInSamplesY shall be less than or equal to MaxLumaPs, where MaxLumaPs is specified in Table A.8 for the tier and level of the layer.

b) The value of pic_width_vps_in_luma_samples of the vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format( ) syntax structure in the VPS shall be less than or equal to Sqrt(MaxLumaPs * 8).

c) The value of pic_height_vps_in_luma_samples of the vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format( ) syntax structure in the VPS shall be less than or equal to Sqrt(MaxLumaPs * 8).

d) The value of max_vps_dec_pic_buffering_minus1[olsIdx][LayerIdxInVps[currLayerId]][HighestTid] shall be less than or equal to MaxDpbSize as derived by Equation A-2, with PicSizeInSamplesY being replaced with layerSizeInSamplesY, for the tier and level of the layer.

e) For level 5 and higher levels, the value of CtbSizeY for the layer shall be equal to 32 or 64.

f) The value of NumPicTotalCurr for each picture in the layer shall be less than or equal to 8.

g) When decoding each coded picture in the layer, the value of num_tile_columns_minus1 shall be less than MaxTileCols and num_tile_rows_minus1 shall be less than MaxTileRows, where MaxTileCols and MaxTileRows are specified in Table A.8 for the tier and level of the layer.

h) For the VCL HRD parameters of the layer, CpbSize[i] shall be less than or equal to CpbVclFactor * MaxCPB for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0][HighestTid], inclusive, where CpbSize[i] is specified in clause F.13.1 and MaxCPB is specified in Table A.8 for the tier and level of the layer in units of CpbVclFactor bits.

i) For the NAL HRD parameters of the layer, CpbSize[i] shall be less than or equal to CpbNalFactor * MaxCPB for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0][HighestTid], inclusive, where CpbSize[i] is specified in clause F.13.1 and MaxCPB is specified in Table A.8 for the tier and level of the layer in units of CpbNalFactor bits.

Table A.8 specifies the limits for each level of each tier for levels other than level 8.5.

A tier and level to which a layer in an output operation point associated with an OLS in a bitstream conforms are indicated by the syntax elements general_tier_flag and general_level_idc if OpTid of the output layer set is equal to vps_max_sub_layer_minus1, and by the syntax elements sub_layer_tier_flag[OpTid] and sub_layer_level_idc[OpTid] otherwise, as follows:

– If the specified level is not level 8.5, general_tier_flag or sub_layer_tier_flag[OpTid] equal to 0 indicates conformance to the Main tier, and general_tier_flag or sub_layer_tier_flag[OpTid] equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8, and general_tier_flag and sub_layer_tier_flag[OpTid] shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A.8 marked with "-"). Otherwise (the specified level is level 8.5), it is a requirement of bitstream conformance that general_tier_flag and sub_layer_tier_flag[OpTid] shall be equal to 1 and the value 0 for general_tier_flag and sub_layer_tier_flag[OpTid] is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of general_tier_flag and sub_layer_tier_flag[OpTid].
– general_level_idc and sub_layer_level_idc[OpTid] shall be set equal to a value of 30 times the level number specified in Table A.8.

G.11.2.2 Profile-specific tier and level limits for the Multiview Main profile

The following is specified for expressing the constraints in this clause:

– The variable HbrFactor is set equal to 1.

– The variable BrVclFactor is set equal to CpbVclFactor * HbrFactor.
– The variable BrNalFactor is set equal to CpbNalFactor * HbrFactor.
– The variable MinCr is set equal to MinCrBase * MinCrScaleFactor ÷ HbrFactor, where MinCrBase is specified in Table A.9.

Each layer conforming to the Multiview Main profile at a specified tier and level shall obey the following constraints for each conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer, and the CPB is understood to be the BPB:

a) The nominal removal time of access unit n (with n greater than 0) from the CPB, as specified in clause F.13.2.3, shall satisfy the constraint that AuNominalRemovalTime[n] − AuCpbRemovalTime[n − 1] is greater than or equal to Max(layerSizeInSamplesY ÷ MaxLumaSr, fR), where layerSizeInSamplesY is the value of layerSizeInSamplesY for access unit n − 1 and MaxLumaSr is the value specified in Table A.9 that applies to access unit n − 1 for the tier and level of the layer.

b) The difference between consecutive output times of pictures in different access units, as specified in clause F.13.3.3, shall satisfy the constraint that DpbOutputInterval[n] is greater than or equal to Max(layerSizeInSamplesY ÷ MaxLumaSr, fR), where layerSizeInSamplesY is the value of layerSizeInSamplesY of access unit n and MaxLumaSr is the value specified in Table A.9 for access unit n for the tier and level of the layer, provided that access unit n is an access unit that has a picture that is output and is not the last of such access units.

c) The removal time of access unit 0 shall satisfy the constraint that the number of coded slice segments in access unit 0 is less than or equal to Min(Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0]) + MaxSliceSegmentsPerPicture * layerSizeInSamplesY / MaxLumaPs), MaxSliceSegmentsPerPicture), for the value of layerSizeInSamplesY of access unit 0, where MaxSliceSegmentsPerPicture, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9 for the tier and level of the layer.

d) The difference between consecutive CPB removal times of access units n and n − 1 (with n greater than 0) shall satisfy the constraint that the number of slice segments in access unit n is less than or equal to Min((Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1]))), MaxSliceSegmentsPerPicture), where MaxSliceSegmentsPerPicture, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9 that apply to access unit n for the tier and level of the layer.

e) For the VCL HRD parameters for the layer, BitRate[i] shall be less than or equal to BrVclFactor * MaxBR for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0][HighestTid], inclusive, where BitRate[i] is specified in clause F.13.1 and MaxBR is specified in Table A.9 in units of BrVclFactor bits/s for the tier and level of the layer.

f) For the NAL HRD parameters for the layer, BitRate[i] shall be less than or equal to BrNalFactor * MaxBR for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0][HighestTid], inclusive, where BitRate[i] is specified in clause F.13.1 and MaxBR is specified in Table A.9 in units of BrNalFactor bits/s for the tier and level of the layer.

g) The sum of the NumBytesInNalUnit variables for access unit 0 shall be less than or equal to FormatCapabilityFactor * (Max(layerSizeInSamplesY, fR * MaxLumaSr) + MaxLumaSr * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0])) ÷ MinCr for the value of layerSizeInSamplesY of access unit 0, where MaxLumaSr is specified in Table A.9, and both MaxLumaSr and FormatCapabilityFactor are the values that apply to access unit 0 for the tier and level of the layer.

h) The sum of the NumBytesInNalUnit variables for access unit n (with n greater than 0) shall be less than or equal to FormatCapabilityFactor * MaxLumaSr * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1]) ÷ MinCr, where MaxLumaSr is specified in Table A.9, and both MaxLumaSr and FormatCapabilityFactor are the values that apply to access unit n for the tier and level of the layer.

i) The removal time of access unit 0 shall satisfy the constraint that the number of tiles in coded pictures in access unit 0 is less than or equal to Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0]) + MaxTileCols * MaxTileRows * PicSizeInSamplesY / MaxLumaPs), MaxTileCols * MaxTileRows), for the value of layerSizeInSamplesY of access unit 0, where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to access unit 0 for the tier and level of the layer.

j) The difference between consecutive CPB removal times of access units n and n − 1 (with n greater than 0) shall satisfy the constraint that the number of tiles in coded pictures in access unit n is less than or equal to Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1])), MaxTileCols * MaxTileRows), where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to access unit n for the tier and level of the layer.

G.11.3 Decoder capabilities

When a decoder conforms to any profile specified in Annex G, it shall also have the INBLD capability specified in clause F.11.1.

Clause F.11.2 specifies requirements for a decoder conforming to any profile specified in Annex G.

G.12 Byte stream format

The specifications in clause F.12 apply.

G.13 Hypothetical reference decoder

The specifications in clause F.13 and its subclauses apply.

G.14 Supplemental enhancement information

G.14.1 General

The specifications in clause F.14.1 apply.

G.14.2 SEI payload syntax

G.14.2.1 General SEI payload syntax

The specifications in clause F.14.2.1 apply.

G.14.2.2 Annex D and Annex F SEI message syntax for multiview high efficiency video coding

The specifications in clauses F.14.2.2 through F.14.2.11 apply.

G.14.2.3 3D reference displays information SEI message syntax

three_dimensional_reference_displays_info(payloadSize) {	**Descriptor**
prec_ref_display_width	ue(v)
ref_viewing_distance_flag	u(1)
if(ref_viewing_distance_flag)	
prec_ref_viewing_dist	ue(v)
num_ref_displays_minus1	ue(v)
for(i = 0; i <= num_ref_displays_minus1; i++) {	
left_view_id[i]	ue(v)
right_view_id[i]	ue(v)
exponent_ref_display_width[i]	u(6)
mantissa_ref_display_width[i]	u(v)
if(ref_viewing_distance_flag) {	
exponent_ref_viewing_distance[i]	u(6)
mantissa_ref_viewing_distance[i]	u(v)
}	
additional_shift_present_flag[i]	u(1)
if(additional_shift_present_flag[i])	
num_sample_shift_plus512[i]	u(10)
}	
three_dimensional_reference_displays_extension_flag	u(1)
}	

G.14.2.4 Depth representation information SEI message syntax

G.14.2.4.1 General

depth_representation_info(payloadSize) {	**Descriptor**
z_near_flag	u(1)
z_far_flag	u(1)
d_min_flag	u(1)
d_max_flag	u(1)
depth_representation_type	ue(v)
if(d_min_flag || d_max_flag)	
disparity_ref_view_id	ue(v)
if(z_near_flag)	
depth_rep_info_element(ZNearSign, ZNearExp, ZNearMantissa, ZNearManLen)	
if(z_far_flag)	
depth_rep_info_element(ZFarSign, ZFarExp, ZFarMantissa, ZFarManLen)	
if(d_min_flag)	
depth_rep_info_element(DMinSign, DMinExp, DMinMantissa, DMinManLen)	
if(d_max_flag)	
depth_rep_info_element(DMaxSign, DMaxExp, DMaxMantissa, DMaxManLen)	
if(depth_representation_type = = 3) {	
depth_nonlinear_representation_num_minus1	ue(v)
for(i = 1; i <= depth_nonlinear_representation_num_minus1 + 1; i++)	
depth_nonlinear_representation_model[i]	ue(v)
}	
}	

G.14.2.4.2 Depth representation information element syntax

depth_rep_info_element(OutSign, OutExp, OutMantissa, OutManLen) {	**Descriptor**
da_sign_flag	u(1)
da_exponent	u(7)
da_mantissa_len_minus1	u(5)
da_mantissa	u(v)
}	

G.14.2.5 Multiview scene information SEI message syntax

multiview_scene_info(payloadSize) {	**Descriptor**
min_disparity	se(v)
max_disparity_range	ue(v)
}	

G.14.2.6 Multiview acquisition information SEI message syntax

multiview_acquisition_info(payloadSize) {	**Descriptor**
intrinsic_param_flag	u(1)
extrinsic_param_flag	u(1)
if(intrinsic_param_flag) {	
intrinsic_params_equal_flag	u(1)
prec_focal_length	ue(v)
prec_principal_point	ue(v)
prec_skew_factor	ue(v)
for(i = 0; i <= intrinsic_params_equal_flag ? 0 : numViewsMinus1; i++) {	
sign_focal_length_x[i]	u(1)
exponent_focal_length_x[i]	u(6)
mantissa_focal_length_x[i]	u(v)
sign_focal_length_y[i]	u(1)
exponent_focal_length_y[i]	u(6)
mantissa_focal_length_y[i]	u(v)
sign_principal_point_x[i]	u(1)
exponent_principal_point_x[i]	u(6)
mantissa_principal_point_x[i]	u(v)
sign_principal_point_y[i]	u(1)
exponent_principal_point_y[i]	u(6)
mantissa_principal_point_y[i]	u(v)
sign_skew_factor[i]	u(1)
exponent_skew_factor[i]	u(6)
mantissa_skew_factor[i]	u(v)
}	
}	
if(extrinsic_param_flag) {	
prec_rotation_param	ue(v)
prec_translation_param	ue(v)
for(i = 0; i <= numViewsMinus1; i++)	
for(j = 0; j < 3; j++) { /* row */	
for(k = 0; k < 3; k++) { /* column */	
sign_r[i][j][k]	u(1)
exponent_r[i][j][k]	u(6)
mantissa_r[i][j][k]	u(v)
}	
sign_t[i][j]	u(1)
exponent_t[i][j]	u(6)
mantissa_t[i][j]	u(v)
}	
}	
}	

G.14.2.7 Multiview view position SEI message syntax

multiview_view_position(payloadSize) {	**Descriptor**
num_views_minus1	ue(v)
for(i = 0; i <= num_views_minus1; i++)	
view_position[i]	ue(v)
}	

G.14.3 SEI payload semantics

G.14.3.1 General SEI payload semantics

The specifications in clause F.14.3.1 apply with the following modifications and additions:

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 161, 165, 167, 168, 177, 178, 179, 200 to 202, inclusive, and 205.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 168, inclusive, 176 to 180, inclusive, 200 to 202, inclusive, and 205.

The semantics and persistence scope for each SEI message specified in this annex are specified in the semantics specification for each particular SEI message in the subclauses of this clause.

NOTE – Persistence information for SEI messages specified in this annex is informatively summarized in Table G.1.

Table G.1 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
3D reference displays information	Specified by the semantics of the SEI message
Depth representation information	Specified by the semantics of the SEI message.
Multiview scene information	The CVS containing the SEI message
Multiview acquisition information	The CVS containing the SEI message
Multiview view position SEI message	The CVS containing the SEI message

G.14.3.2 Annex D and Annex F SEI message semantics for multiview high efficiency video coding

G.14.3.2.1 General

The specifications of clause F.14.3.2 and its subclauses apply with the modifications specified in clause G.14.3.2.2.

G.14.3.2.2 Scalable nesting SEI message semantics for multiview high efficiency video coding

The specifications of clause F.14.3.2.7 apply with the following additions:

An SEI message that has payloadType equal to 176 (3D reference displays information) or 180 (multiview view position) shall not be directly contained in a scalable nesting SEI message.

When the scalable nesting SEI message contains an SEI message that has payloadType equal to 177, 178 or 179, bitstream_subset_flag shall be equal to 0.

G.14.3.2.3 3D reference displays information SEI message semantics

A 3D reference displays information SEI message contains information about the reference display width(s) and reference viewing distance(s) as well as information about the corresponding reference stereo pair(s), i.e., the pair(s) of views to be displayed for the viewer's left and right eyes on the reference display at the reference viewing distance. This information enables a view renderer to generate a proper stereo pair for the target screen width and the viewing distance. The reference display width and viewing distance values are signalled in units of centimetres. The reference pair of view specified in this SEI message can be used to extract or infer parameters related to the distance between the camera centres in the reference stereo pair, which can be used for generation of views for the target display. For multi-view displays, the reference stereo pair corresponds to a pair of views that can be simultaneously observed by the viewer's left and right eyes.

When present, this SEI message shall be associated with an IRAP access unit or with a non-IRAP access unit, when all access units that follow this access unit in the decoding order also follow it in output order. The 3D reference display

information SEI message applies to the current access unit and all the access units which follow this access unit in both the output and decoding order until but not including the next IRAP access unit or the next access unit containing a 3D reference displays information SEI message.

NOTE 1 – The 3D reference displays information SEI message specifies display parameters for which the 3D sequence was optimized and the corresponding reference parameters. Each reference display (i.e., a reference display width and possibly a corresponding viewing distance) is associated with one reference pair of views by signalling their ViewId. The difference between the values of ViewId is referred to as the baseline distance (i.e., the distance between the centres of the cameras used to obtain the video sequence).

The following equations can be used for determining the baseline distance and horizontal shift for the receiver's display when the ratio between the receiver's viewing distance and the reference viewing distance is the same as the ratio between the receiver screen width and the reference screen width:

$$\text{baseline}[\ i\ ] = \text{refBaseline}[\ i\ ] * (\ \text{refDisplayWidth}[\ i\ ] \div \text{displayWidth}\ ) \qquad \text{(G-3)}$$

$$\text{shift}[\ i\ ] = \text{refShift}[\ i\ ] * (\ \text{refDisplayWidth}[\ i\ ] \div \text{displayWidth}\ ) \qquad \text{(G-4)}$$

where refBaseline[i] is equal to right_view_id[i] − left_view_id[i] signalled in this SEI message. Other parameters related to the view generation may be obtained determined by using a similar equation.

$$\text{parameter}[\ i\ ] = \text{refParameter}[\ i\ ] * (\ \text{refDisplayWidth}[\ i\ ] \div \text{displayWidth}\ ) \qquad \text{(G-5)}$$

where refParameter[i] is a parameter related to view generation that corresponds to the reference pair of views signalled by left_view_id[i] and right_view_id[i]. In the above equations, the width of the visible part of the display used for showing the video sequence should be understood under "display width". The same equations can also be used for determining the pair of views and horizontal shift or other view synthesis parameters when the viewing distance is not scaled proportionally to the screen width compared to the reference display parameters. In this case, the effect of applying the above equations would be to keep the perceived depth in the same proportion to the viewing distance as in the reference setup.

When the view synthesis related parameters that correspond to the reference stereo pair change from one access unit to another, they should be scaled with the same scaling factor as the parameters in the access unit that the SEI message is associated with. Therefore, the above equation should also be applied to obtain the parameters for a following access unit, where the refParameter is the parameter related to the reference stereo pair associated the following access unit.

The horizontal shift for the receiver's display should also be modified by scaling it with the same factor as that used to scale the baseline distance (or other view synthesis parameters).

prec_ref_display_width specifies the exponent of the maximum allowable truncation error for refDisplayWidth[i] as given by $2^{-\text{prec\_ref\_display\_width}}$. The value of prec_ref_display_width shall be in the range of 0 to 31, inclusive.

ref_viewing_distance_flag equal to 1 indicates the presence of reference viewing distance. ref_viewing_distance_flag equal to 0 indicates that the reference viewing distance is not present.

prec_ref_viewing_dist specifies the exponent of the maximum allowable truncation error for refViewingDist[i] as given by $2^{-\text{prec\_ref\_viewing\_dist}}$. The value of prec_ref_viewing_dist shall be in the range of 0 to 31, inclusive.

num_ref_displays_minus1 plus 1 specifies the number of reference displays that are signalled in this SEI message. The value of num_ref_displays_minus1 shall be in the range of 0 to 31, inclusive.

left_view_id[i] indicates the ViewId of the left view of a stereo pair corresponding to the i-th reference display.

right_view_id[i] indicates the ViewId of the right view of a stereo-pair corresponding to the i-th reference display.

exponent_ref_display_width[i] specifies the exponent part of the reference display width of the i-th reference display. The value of exponent_ref_display_width[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified reference display width.

mantissa_ref_display_width[i] specifies the mantissa part of the reference display width of the i-th reference display. The variable refDispWidthBits specifying the number of bits of the mantissa_ref_display_width[i] syntax element is derived as follows:

– If exponent_ref_display_width[i] is equal to 0, refDispWidthBits is set equal to Max(0, prec_ref_display_width − 30).

– Otherwise (0 < exponent_ref_display_width[i] < 63), refDispWidthBits is set equal to Max(0, exponent_ref_display_width[i] + prec_ref_display_width − 31).

exponent_ref_viewing_distance[i] specifies the exponent part of the reference viewing distance of the i-th reference display. The value of exponent_ref_viewing_distance[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified reference display width.

mantissa_ref_viewing_distance[i] specifies the mantissa part of the reference viewing distance of the i-th reference display. The variable refViewDistBits specifying the number of bits of the mantissa_ref_viewing_distance[i] syntax element is derived as follows:

– If exponent_ref_viewing_distance[i] is equal to 0, the refViewDistBits is set equal to Max(0, prec_ref_viewing_distance − 30).

– Otherwise (0 < exponent_ref_viewing_distance[i] < 63), refViewDistBits is set equal to Max(0, exponent_ref_viewing_distance[i] + prec_ref_viewing_distance − 31).

The variables in the x row of Table G.2 are derived from the respective variables or values in the e, n and v rows of Table G.2 as follows:

– If e is not equal to 0, the following applies:

$$x = 2^{(e - 31)} * (1 + n \div 2^{v}) \tag{G-6}$$

– Otherwise (e is equal to 0), the following applies:

$$x = 2^{-(30 + v)} * n \tag{G-7}$$

NOTE 2 – The above specification is similar to that found in IEC 60559:1989.

Table G.2 – Association between camera parameter variables and syntax elements

x	refDisplayWidth[i]	refViewingDistance[i]
e	exponent_ref_display_width[i]	exponent_ref_viewing_distance[i]
n	mantissa_ref_display_width[i]	mantissa_ref_viewing_distance[i]
v	refDispWidthBits	refViewDistBits

additional_shift_present_flag[i] equal to 1 indicates that the information about additional horizontal shift of the left and right views for the i-th reference display is present in this SEI message. additional_shift_present_flag[i] equal to 0 indicates that the information about additional horizontal shift of the left and right views for the i-th reference display is not present in this SEI message.

num_sample_shift_plus512[i] indicates the recommended additional horizontal shift for a stereo pair corresponding to the i-th reference baseline and the i-th reference display.

– If num_sample_shift_plus512[i] is less than 512, it is recommended that the left view of the stereo pair corresponding to the i-th reference baseline and the i-th reference display is shifted in the left direction by (512 − num_sample_shift_plus512[i]) samples with respect to the right view of the stereo pair.

– Otherwise, if num_sample_shift_plus512[i] is equal to 512, it is recommended that shifting is not applied.

– Otherwise, (num_sample_shift_plus512[i] is greater than 512), it is recommended that the left view in the stereo pair corresponding to the i-th reference baseline and the i-th reference display should be shifted in the right direction by (num_sample_shift_plus512[i] − 512) samples with respect to the right view of the stereo pair.

The value of num_sample_shift_plus512[i] shall be in the range of 0 to 1 023, inclusive.

NOTE 3 – Shifting the left view in the left (or right) direction by x samples with respect to the right view can be performed by the following two-step processing:

1) Shift the left view by x / 2 samples in the left (or right) direction and shift the right view by x / 2 samples in the right (or left) direction.

2) Fill the left and right image margins of x / 2 samples in width in both the left and right views in background colour.

The following explains the recommended shifting processing in the case of shifting the left view in the left direction by x samples with respect to the right view:

```
for( i = x / 2; i < width − x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ i ] = leftView[ j ][ i + x / 2 ]
        rightView[ j ][ width − 1 − i ] = rightView[ j ][ width − 1 − i − x / 2 ]        (G-8)
    }
for( i = 0; i < x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ width − 1 − i ] = leftView[ j ][ i ] = backgroundColour
        rightView[ j ][ width − 1 − i ] = rightView[ j ][ i ] = backgroundColour
    }
```

The following explains the recommended shifting processing in the case of shifting the left view in the right direction by x samples with respect to the right view:

```
for( i = x / 2; i < width − x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ width − 1 − i ] = leftView[ j ][ width − 1 − i − x / 2 ]
        rightView[ j ][ i ] = rightView[ j ][ i + x / 2 ]                          (G-9)
    }
for( i = 0; i < x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ width − 1 − i ] = leftView[ j ][ i ] = backgroundColour
        rightView[ j ][ width − 1 − i ] = rightView[ j ][ i ] = backgroundColour
    }
```

The variable backgroundColour may take different values in different systems, for example black or grey.

three_dimensional_reference_displays_extension_flag equal to 0 indicates that no additional data follows within the reference displays SEI message. The value of three_dimensional_reference_displays_extension_flag shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for three_dimensional_reference_displays_extension_flag is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all data that follow the value 1 for three_dimensional_reference_displays_extension_flag in a reference displays SEI message.

G.14.3.3 Depth representation information SEI message semantics

G.14.3.3.1 General

The syntax elements in the depth representation information SEI message specify various parameters for auxiliary pictures of type AUX_DEPTH for the purpose of processing decoded primary and auxiliary pictures prior to rendering on a 3D display, such as view synthesis. Specifically, depth or disparity ranges for depth pictures are specified.

When present, the depth representation information SEI message shall be associated with one or more layers with AuxId value equal to AUX_DEPTH. The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the depth representation information SEI message applies.

When present, the depth representation information SEI message may be included in any access unit. It is recommended that, when present, the SEI message is included for the purpose of random access in an access unit in which the coded picture with nuh_layer_id equal to targetLayerId is an IRAP picture.

For an auxiliary picture with AuxId[targetLayerId] equal to AUX_DEPTH, an associated primary picture, if any, is a picture in the same access unit having AuxId[nuhLayerIdB] equal to 0 such that ScalabilityId[LayerIdxInVps[targetLayerId]][j] is equal to ScalabilityId[LayerIdxInVps[nuhLayerIdB]][j] for all values of j in the range of 0 to 2, inclusive, and 4 to 15, inclusive.

The information indicated in the SEI message applies to all the pictures with nuh_layer_id equal to targetLayerId from the access unit containing the SEI message up to but excluding the next picture, in decoding order, associated with a depth representation information SEI message applicable to targetLayerId or to the end of the CLVS of the nuh_layer_id equal to targetLayerId, whichever is earlier in decoding order.

z_near_flag equal to 0 specifies that the syntax elements specifying the nearest depth value are not present in the syntax structure. z_near_flag equal to 1 specifies that the syntax elements specifying the nearest depth value are present in the syntax structure.

z_far_flag equal to 0 specifies that the syntax elements specifying the farthest depth value are not present in the syntax structure. z_far_flag equal to 1 specifies that the syntax elements specifying the farthest depth value are present in the syntax structure.

d_min_flag equal to 0 specifies that the syntax elements specifying the minimum disparity value are not present in the syntax structure. d_min_flag equal to 1 specifies that the syntax elements specifying the minimum disparity value are present in the syntax structure.

d_max_flag equal to 0 specifies that the syntax elements specifying the maximum disparity value are not present in the syntax structure. d_max_flag equal to 1 specifies that the syntax elements specifying the maximum disparity value are present in the syntax structure.

depth_representation_type specifies the representation definition of decoded luma samples of auxiliary pictures as specified in Table G.3. In Table G.3, disparity specifies the horizontal displacement between two texture views and Z value specifies the distance from a camera.

The variable maxVal is set equal to (1 << (8 + bit_depth_luma_minus8)) − 1, where bit_depth_luma_minus8 is the value included in or inferred for the active SPS of the layer with nuh_layer_id equal to targetLayerId. The value of

depth_representation_type shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this version of this Specification. The values of 4 to 15, inclusive, for depth_representation_type are reserved for future use by ITU-T | ISO/IEC. Although the value of depth_representation_type is required to be in the range of 0 to 3, inclusive, in this version of this Specification, decoders shall allow values of depth_representation_type in the range of 4 to 15, inclusive, to appear in the syntax. Decoders conforming to this version of this Specification shall ignore all data that follow a value of depth_representation_type in the range of 4 to 15, inclusive, in the depth representation information SEI messsage.

Table G.3 – Definition of depth_representation_type

depth_representation_type	Interpretation
0	Each decoded luma sample value of an auxiliary picture represents an inverse of Z value that is uniformly quantized into the range of 0 to maxVal, inclusive. When z_far_flag is equal to 1, the luma sample value equal to 0 represents the inverse of ZFar (specified below). When z_near_flag is equal to 1, the luma sample value equal to maxVal represents the inverse of ZNear (specified below).
1	Each decoded luma sample value of an auxiliary picture represents disparity that is uniformly quantized into the range of 0 to maxVal, inclusive. When d_min_flag is equal to 1, the luma sample value equal to 0 represents DMin (specified below). When d_max_flag is equal to 1, the luma sample value equal to maxVal represents DMax (specified below).
2	Each decoded luma sample value of an auxiliary picture represents a Z value uniformly quantized into the range of 0 to maxVal, inclusive. When z_far_flag is equal to 1, the luma sample value equal to 0 corresponds to ZFar (specified below). When z_near_flag is equal to 1, the luma sample value equal to maxVal represents ZNear (specified below).
3	Each decoded luma sample value of an auxiliary picture represents a non-linearly mapped disparity, normalized in range from 0 to maxVal, as specified by depth_nonlinear_representation_num_minus1 and depth_nonlinear_representation_model[i]. When d_min_flag is equal to 1, the luma sample value equal to 0 represents DMin (specified below). When d_max_flag is equal to 1, the luma sample value equal to maxVal represents DMax (specified below).
4..15	Reserved

disparity_ref_view_id specifies the ViewId value for which the disparity values are derived. The value of disparity_ref_view_id shall be in the range of 0 to 1 023, inclusive.

NOTE 1 – disparity_ref_view_id is present only if d_min_flag is equal to 1 or d_max_flag is equal to 1 and is useful for depth_representation_type values equal to 1 and 3.

The variables in the x column of Table G.4 are derived from the respective variables in the s, e, n and v columns of Table G.4 as follows:

– If the value of e is in the range of 0 to 127, exclusive, x is set equal to $(-1)^{s} * 2^{e-31} * (1 + n \div 2^{v})$.

– Otherwise (e is equal to 0), x is set equal to $(-1)^{s} * 2^{-(30 + v)} * n$.

NOTE 1 – The above specification is similar to that found in IEC 60559:1989.

Table G.4 – Association between depth parameter variables and syntax elements

x	S	e	n	v
ZNear	ZNearSign	ZNearExp	ZNearMantissa	ZNearManLen
ZFar	ZFarSign	ZFarExp	ZFarMantissa	ZFarManLen
DMax	DMaxSign	DMaxExp	DMaxMantissa	DMaxManLen
DMin	DMinSign	DMinExp	DMinMantissa	DMinManLen

The DMin and DMax values, when present, are specified in units of a luma sample width of the coded picture with ViewId equal to ViewId of the auxiliary picture.

The units for the ZNear and ZFar values, when present, are identical but unspecified.

depth_nonlinear_representation_num_minus1 plus 2 specifies the number of piece-wise linear segments for mapping of depth values to a scale that is uniformly quantized in terms of disparity. The value of depth_nonlinear_representation_num_minus1 shall be in the range of 0 to 62, inclusive.

depth_nonlinear_representation_model[i] for i ranging from 0 to depth_nonlinear_representation_num_minus1 + 2, inclusive, specify the piece-wise linear segments for mapping of decoded luma sample values of an auxiliary picture to a scale that is uniformly quantized in terms of disparity. The value of depth_nonlinear_representation_model[i] shall be in the range of 0 to 65 535, inclusive. The values of depth_nonlinear_representation_model[0] and depth_nonlinear_representation_model[depth_nonlinear_representation_num_minus1 + 2] are both inferred to be equal to 0.

NOTE 2 – When depth_representation_type is equal to 3, an auxiliary picture contains non-linearly transformed depth samples. The variable DepthLUT[i], as specified below, is used to transform decoded depth sample values from the non-linear representation to the linear representation, i.e., uniformly quantized disparity values. The shape of this transform is defined by means of line-segment approximation in two-dimensional linear-disparity-to-non-linear-disparity space. The first (0, 0) and the last (maxVal, maxVal) nodes of the curve are predefined. Positions of additional nodes are transmitted in form of deviations (depth_nonlinear_representation_model[i]) from the straight-line curve. These deviations are uniformly distributed along the whole range of 0 to maxVal, inclusive, with spacing depending on the value of nonlinear_depth_representation_num_minus1.

The variable DepthLUT[i] for i in the range of 0 to maxVal, inclusive, is specified as follows:

```
for( k = 0; k  <=  depth_nonlinear_representation_num_minus1 + 1; k++ ) {
    pos1 = ( maxVal * k ) / (depth_nonlinear_representation_num_minus1 + 2 )
    dev1 = depth_nonlinear_representation_model[ k ]
    pos2 = ( maxVal * ( k + 1 ) ) / (depth_nonlinear_representation_num_minus1 + 2 )
    dev2 = depth_nonlinear_representation_model[ k + 1 ]                                  (G-10)

    x1 = pos1 − dev1
    y1 = pos1 + dev1
    x2 = pos2 − dev2
    y2 = pos2 + dev2

    for( x = Max( x1, 0 ); x  <=  Min( x2, maxVal ); x++ )
        DepthLUT[ x ] = Clip3( 0, maxVal, Round( ( ( x − x1 ) * ( y2 − y1 ) ) ÷ ( x2 − x1 ) + y1 ) )
}
```

When depth_representation_type is equal to 3, DepthLUT[dS] for all decoded luma sample values dS of an auxiliary picture in the range of 0 to maxVal, inclusive, represents disparity that is uniformly quantized into the range of 0 to maxVal, inclusive.

G.14.3.3.2 Depth representation information element semantics

The syntax structure specifies the value of an element in the depth representation information SEI message.

The depth_rep_info_element(OutSign, OutExp, OutMantissa, OutManLen) syntax structure sets the values of the OutSign, OutExp, OutMantissa and OutManLen variables that represent a floating-point value. When the syntax structure is included in another syntax structure, the variable names OutSign, OutExp, OutMantissa and OutManLen are to be interpreted as being replaced by the variable names used when the syntax structure is included.

da_sign_flag equal to 0 indicates that the sign of the floating-point value is positive. da_sign_flag equal to 1 indicates that the sign is negative. The variable OutSign is set equal to da_sign_flag.

da_exponent specifies the exponent of the floating-point value. The value of da_exponent shall be in the range of 0 to $2^7 - 2$, inclusive. The value $2^7 - 1$ is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value $2^7 - 1$ as indicating an unspecified value. The variable OutExp is set equal to da_exponent.

da_mantissa_len_minus1 plus 1 specifies the number of bits in the da_mantissa syntax element. The variable OutManLen is set equal to da_mantissa_len_minus1 + 1.

da_mantissa specifies the mantissa of the floating-point value. The variable OutMantissa is set equal to da_mantissa.

G.14.3.4 Multiview scene information SEI message semantics

The multiview scene information SEI message indicates the minimum disparity and the maximum disparity among multiple views in an access unit. The minimum disparity and the maximum disparity could be used for processing the decoded views prior to rendering on a 3D display. When present, the multiview scene information SEI message shall be associated with an IRAP access unit. The information signalled in the SEI message applies to the coded video sequence. If the SEI message is not contained within a scalable nesting SEI message, it applies to the views with nuh_layer_id greater than or equal to the nuh_layer_id value of the SEI NAL unit that contains the SEI message. Otherwise (the SEI message is contained within a scalable nesting SEI message), the SEI message applies to the view with nuh_layer_id values identified in the scalable nesting SEI message. The views to which the SEI message applies are referred to as applicable views below.

The actual minimum disparity value may be greater than the one signalled in the multiview scene information SEI message and the actual maximum disparity value may be less than the one signalled in the multiview scene information SEI message, due to that some views in the coded video sequence may have been removed from the original bitstream to produce an extracted sub-bitstream, for example according to the process specified in clause 10.

Let xR be a luma sample position within a luma sample array of the right-side picture of any spatially adjacent views among the applicable views in an access unit. Let xL be the respective luma sample position within a luma sample array of the left-side picture of the same spatially adjacent view such that sample in the luma sample position xL in the left-side view represents the same content as luma sample position xR in the right-side picture. When pic_width_in_luma_samples for the left-side picture, picWidthL, is not equal to pic_width_in_luma_samples for the right-side picture, picWidthR, xL is normalized to the horizontal luma sample resolution of the right-side picture, i.e., xL is set equal to Round(xL * (picWidthR ÷ picWidthL)). The disparity between xR and xL is specified to be equal to (xR − xL). The maximum disparity between two pictures is defined to be the maximum of the disparity between any sample position pairs xR and xL. The minimum disparity between two pictures is defined to be the minimum of the disparity between any sample position pairs xR and xL.

min_disparity specifies the minimum disparity, in units of luma samples, between pictures of any spatially adjacent views among the applicable views in an access unit. The value of min_disparity shall be in the range of −1 024 to 1 023, inclusive.

max_disparity_range specifies that the maximum disparity, in units of luma samples, between pictures of any spatially adjacent views among the applicable views in an access unit. The value of max_disparity_range shall be in the range of 0 to 2047, inclusive.

NOTE – The minimum disparity and the maximum disparity depend on the baseline distance between spatially adjacent views and the spatial resolution of each view. Therefore, if either the number of views or spatial resolution is changed, the minimum disparity and the maximum disparity should also be changed accordingly.

G.14.3.5 Multiview acquisition information SEI message semantics

The multiview acquisition information SEI message specifies various parameters of the acquisition environment. Specifically, intrinsic and extrinsic camera parameters are specified. These parameters could be used for processing the decoded views prior to rendering on a 3D display.

The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the multiview acquisition information SEI message applies as specified in clause D.3.1.

When present, the multiview acquisition information SEI message that applies to the current layer shall be included in an access unit that contains an IRAP picture that is the first picture of a CLVS of the current layer. The information signalled in the SEI message applies to the CLVS.

When the multiview acquisition information SEI message is included in a scalable nesting SEI message, the syntax elements bitstream_subset_flag, nesting_op_flag and all_layers_flag in the scalable nesting SEI message shall be equal to 0.

The variable numViewsMinus1 is derived as follows:

– If the multiview acquisition information SEI message is not included in a scalable nesting SEI message, numViewsMinus1 is set equal to 0.

– Otherwise (the multiview acquisition information SEI message is included in a scalable nesting SEI message), numViewsMinus1 is set equal to nesting_num_layers_minus1.

Some of the views for which the multiview acquisition information is included in a multiview acquisition information SEI message may not be present.

In the semantics below, index i refers to the syntax elements and variables that apply to the layer with nuh_layer_id equal to nestingLayerIdList[0][i].

The extrinsic camera parameters are specified according to a right-handed coordinate system, where the upper left corner of the image is the origin, i.e., the (0, 0) coordinate, with the other corners of the image having non-negative coordinates. With these specifications, a 3-dimensional world point, wP = [x y z] is mapped to a 2-dimensional camera point, cP[i] = [u v 1], for the i-th camera according to:

$$s * cP[i] = A[i] * R^{-1}[i] * (wP - T[i]) \qquad \text{(G-11)}$$

where A[i] denotes the intrinsic camera parameter matrix, R^{-1}[i] denotes the inverse of the rotation matrix R[i], T[i] denotes the translation vector and s (a scalar value) is an arbitrary scale factor chosen to make the third coordinate of cP[i] equal to 1. The elements of A[i], R[i] and T[i] are determined according to the syntax elements signalled in this SEI message and as specified below.

intrinsic_param_flag equal to 1 indicates the presence of intrinsic camera parameters. intrinsic_param_flag equal to 0

indicates the absence of intrinsic camera parameters.

extrinsic_param_flag equal to 1 indicates the presence of extrinsic camera parameters. extrinsic_param_flag equal to 0 indicates the absence of extrinsic camera parameters.

intrinsic_params_equal_flag equal to 1 indicates that the intrinsic camera parameters are equal for all cameras and only one set of intrinsic camera parameters is present. intrinsic_params_equal_flag equal to 0 indicates that the intrinsic camera parameters are different for each camera and that a set of intrinsic camera parameters is present for each camera.

prec_focal_length specifies the exponent of the maximum allowable truncation error for focal_length_x[i] and focal_length_y[i] as given by $2^{-\text{prec\_focal\_length}}$. The value of prec_focal_length shall be in the range of 0 to 31, inclusive.

prec_principal_point specifies the exponent of the maximum allowable truncation error for principal_point_x[i] and principal_point_y[i] as given by $2^{-\text{prec\_principal\_point}}$. The value of prec_principal_point shall be in the range of 0 to 31, inclusive.

prec_skew_factor specifies the exponent of the maximum allowable truncation error for skew factor as given by $2^{-\text{prec\_skew\_factor}}$. The value of prec_skew_factor shall be in the range of 0 to 31, inclusive.

sign_focal_length_x[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the horizontal direction is positive. sign_focal_length_x[i] equal to 1 indicates that the sign is negative.

exponent_focal_length_x[i] specifies the exponent part of the focal length of the i-th camera in the horizontal direction. The value of exponent_focal_length_x[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

mantissa_focal_length_x[i] specifies the mantissa part of the focal length of the i-th camera in the horizontal direction. The length of the mantissa_focal_length_x[i] syntax element in units of bits is variable and determined as follows:

– If exponent_focal_length_x[i] is equal to 0, the length is Max(0, prec_focal_length − 30).

– Otherwise (exponent_focal_length_x[i] is in the range of 0 to 63, exclusive), the length is Max(0, exponent_focal_length_x[i] + prec_focal_length − 31).

sign_focal_length_y[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the vertical direction is positive. sign_focal_length_y[i] equal to 1 indicates that the sign is negative.

exponent_focal_length_y[i] specifies the exponent part of the focal length of the i-th camera in the vertical direction. The value of exponent_focal_length_y[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

mantissa_focal_length_y[i] specifies the mantissa part of the focal length of the i-th camera in the vertical direction.

The length of the mantissa_focal_length_y[i] syntax element in units of bits is variable and determined as follows:

– If exponent_focal_length_y[i] is equal to 0, the length is Max(0, prec_focal_length − 30).

– Otherwise (exponent_focal_length_y[i] is in the range of 0 to 63, exclusive), the length is Max(0, exponent_focal_length_y[i] + prec_focal_length − 31).

sign_principal_point_x[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the horizontal direction is positive. sign_principal_point_x[i] equal to 1 indicates that the sign is negative.

exponent_principal_point_x[i] specifies the exponent part of the principal point of the i-th camera in the horizontal direction. The value of exponent_principal_point_x[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified principal point.

mantissa_principal_point_x[i] specifies the mantissa part of the principal point of the i-th camera in the horizontal direction. The length of the mantissa_principal_point_x[i] syntax element in units of bits is variable and is determined as follows:

– If exponent_principal_point_x[i] is equal to 0, the length is Max(0, prec_principal_point − 30).

– Otherwise (exponent_principal_point_x[i] is in the range of 0 to 63, exclusive), the length is Max(0, exponent_principal_point_x[i] + prec_principal_point − 31).

sign_principal_point_y[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the vertical direction is positive. sign_principal_point_y[i] equal to 1 indicates that the sign is negative.

exponent_principal_point_y[i] specifies the exponent part of the principal point of the i-th camera in the vertical direction. The value of exponent_principal_point_y[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified principal point.

mantissa_principal_point_y[i] specifies the mantissa part of the principal point of the i-th camera in the vertical direction. The length of the mantissa_principal_point_y[i] syntax element in units of bits is variable and is determined as follows:

– If exponent_principal_point_y[i] is equal to 0, the length is Max(0, prec_principal_point − 30).

– Otherwise (exponent_principal_point_y[i] is in the range of 0 to 63, exclusive), the length is Max(0, exponent_principal_point_y[i] + prec_principal_point − 31).

sign_skew_factor[i] equal to 0 indicates that the sign of the skew factor of the i-th camera is positive.

sign_skew_factor[i] equal to 1 indicates that the sign is negative.

exponent_skew_factor[i] specifies the exponent part of the skew factor of the i-th camera. The value of exponent_skew_factor[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified skew factor.

mantissa_skew_factor[i] specifies the mantissa part of the skew factor of the i-th camera. The length of the mantissa_skew_factor[i] syntax element in units of bits is variable and determined as follows:

– If exponent_skew_factor[i] is equal to 0, the length is Max(0, prec_skew_factor − 30).

– Otherwise (exponent_skew_factor[i] is in the range of 0 to 63, exclusive), the length is Max(0, exponent_skew_factor[i] + prec_skew_factor − 31).

The intrinsic matrix A[i] for i-th camera is represented by

$$\begin{bmatrix} \text{focalLengthX[i]} & \text{skewFactor[i]} & \text{principalPointX[i]} \\ 0 & \text{focalLengthY[i]} & \text{principalPointY[i]} \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{(G-12)}$$

prec_rotation_param specifies the exponent of the maximum allowable truncation error for r[i][j][k] as given by $2^{-\text{prec\_rotation\_param}}$. The value of prec_rotation_param shall be in the range of 0 to 31, inclusive.

prec_translation_param specifies the exponent of the maximum allowable truncation error for t[i][j] as given by $2^{-\text{prec\_translation\_param}}$. The value of prec_translation_param shall be in the range of 0 to 31, inclusive.

sign_r[i][j][k] equal to 0 indicates that the sign of (j, k) component of the rotation matrix for the i-th camera is positive. sign_r[i][j][k] equal to 1 indicates that the sign is negative.

exponent_r[i][j][k] specifies the exponent part of (j, k) component of the rotation matrix for the i-th camera. The value of exponent_r[i][j][k] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified rotation matrix.

mantissa_r[i][j][k] specifies the mantissa part of (j, k) component of the rotation matrix for the i-th camera. The length of the mantissa_r[i][j][k] syntax element in units of bits is variable and determined as follows:

– If exponent_r[i] is equal to 0, the length is Max(0, prec_rotation_param − 30).

– Otherwise (exponent_r[i] is in the range of 0 to 63, exclusive), the length is Max(0, exponent_r[i] + prec_rotation_param − 31).

The rotation matrix R[i] for i-th camera is represented as follows:

$$\begin{bmatrix} \text{rE[i][0][0]} & \text{rE[i][0][1]} & \text{rE[i][0][2]} \\ \text{rE[i][1][0]} & \text{rE[i][1][1]} & \text{rE[i][1][2]} \\ \text{rE[i][2][0]} & \text{rE[i][2][1]} & \text{rE[i][2][2]} \end{bmatrix} \qquad \text{(G-13)}$$

sign_t[i][j] equal to 0 indicates that the sign of the j-th component of the translation vector for the i-th camera is positive. sign_t[i][j] equal to 1 indicates that the sign is negative.

exponent_t[i][j] specifies the exponent part of the j-th component of the translation vector for the i-th camera. The value of exponent_t[i][j] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified translation vector.

mantissa_t[i][j] specifies the mantissa part of the j-th component of the translation vector for the i-th camera. The length v of the mantissa_t[i][j] syntax element in units of bits is variable and is determined as follows:

– If exponent_t[i] is equal to 0, the length v is set equal to Max(0, prec_translation_param − 30).

– Otherwise (0 < exponent_t[i] < 63), the length v is set equal to Max(0, exponent_t[i] + prec_translation_param − 31).

The translation vector T[i] for the i-th camera is represented by:

$$\begin{bmatrix} tE[\ i\ ][\ 0\ ] \\ tE[\ i\ ][\ 1\ ] \\ tE[\ i\ ][\ 2\ ] \end{bmatrix} \qquad \text{(G-14)}$$

The association between the camera parameter variables and corresponding syntax elements is specified by Table G.5. Each component of the intrinsic and rotation matrices and the translation vector is obtained from the variables specified in Table G.5 as the variable x computed as follows:

– If e is in the range of 0 to 63, exclusive, x is set equal to $(-1)^{s} * 2^{e-31} * (1 + n \div 2^{v})$.

– Otherwise (e is equal to 0), x is set equal to $(-1)^{s} * 2^{-(30 + v)} * n$.

NOTE – The above specification is similar to that found in IEC 60559:1989.

Table G.5 – Association between camera parameter variables and syntax elements.

x	s	e	n
focalLengthX[i]	sign_focal_length_x[i]	exponent_focal_length_x[i]	mantissa_focal_length_x[i]
focalLengthY[i]	sign_focal_length_y[i]	exponent_focal_length_y[i]	mantissa_focal_length_y[i]
principalPointX[i]	sign_principal_point_x[i]	exponent_principal_point_x[i]	mantissa_principal_point_x[i]
principalPointY[i]	sign_principal_point_y[i]	exponent_principal_point_y[i]	mantissa_principal_point_y[i]
skewFactor[i]	sign_skew_factor[i]	exponent_skew_factor[i]	mantissa_skew_factor[i]
rE[i][j][k]	sign_r[i][j][k]	exponent_r[i][j][k]	mantissa_r[i][j][k]
tE[i][j]	sign_t[i][j]	exponent_t[i][j]	mantissa_t[i][j]

G.14.3.6 Multiview view position SEI message semantics

The multiview view position SEI message specifies the relative view position along a single horizontal axis of views within a CVS. When present, the multiview view position SEI message shall be associated with an IRAP access unit. The information signalled in this SEI message applies to the entire CVS.

num_views_minus1 plus 1 shall be equal to NumViews derived from the active VPS for the CVS. The value of num_views_minus1 shall be in the range of 0 to 62, inclusive.

view_position[i] indicates the order of the view with ViewOrderIdx equal to i among all the views from left to right for the purpose of display, with the order for the left-most view being equal to 0 and the value of the order increasing by 1 for next view from left to right. The value of view_position[i] shall be in the range of 0 to 62, inclusive.

G.15 Video usability information

The specifications in clause F.15 and its subclauses apply.

Annex H

Scalable high efficiency video coding

(This annex forms an integral part of this Recommendation | International Standard.)

H.1 Scope

This annex specifies syntax, semantics and decoding processes for scalable high efficiency video coding that use the syntax, semantics, and decoding process specified in clauses 2-10 and Annexes A-F. This annex also specifies profiles, tier and levels for scalable high efficiency video coding.

H.2 Normative references

The list of normative references in clause F.2 applies.

H.3 Definitions

The specifications in clause F.3 and its subclauses apply.

H.4 Abbreviations

The specifications in clause F.4 apply.

H.5 Conventions

The specifications in clause F.5 apply.

H.6 Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships

The specifications in clause F.6 apply.

H.7 Syntax and semantics

The specifications in clause F.7 and its subclauses apply.

H.8 Decoding processes

H.8.1 General decoding process

H.8.1.1 General

The specifications of clause F.8.1.1 apply.

H.8.1.2 Decoding process for a coded picture with nuh_layer_id greater than 0

The decoding process for the current picture CurrPic is as follows:

1. The decoding of NAL units is specified in clause H.8.2.
2. The processes in clauses H.8.1.3 and H.8.3.4 specify the following decoding processes using syntax elements in the slice segment layer and above:
   – At the beginning of the decoding process for the first slice of the current picture, the process specified in clause H.8.1.3 is invoked.
   – At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause H.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
3. The processes in clauses H.8.4, H.8.5, H.8.6 and H.8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments, and the division of the slice segments into CTUs each form a partitioning of the picture.

H.8.1.3 Decoding process for inter-layer reference picture set

Outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1.

The variable currLayerId is set equal to nuh_layer_id of the current picture.

The variables NumRefLayerPicsProcessing, NumRefLayerPicsSampleProcessing and NumRefLayerPicsMotionProcessing are set equal to 0.

The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 and the following applies:

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
   refPicSet0Flag =
      ( ( ViewId[ currLayerId ]  <=  ViewId[ 0 ]  &&ViewId[ currLayerId ]  <=
ViewId[ RefPicLayerId[ i ] ] )  | |
      ( ViewId[ currLayerId ]  >=  ViewId[ 0 ]  &&  ViewId[ currLayerId ]  >=
ViewId[ RefPicLayerId[ i ] ] ) )
   if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
            nuh_layer_id equal to RefPicLayerId[ i ] ) {
      an inter-layer reference picture ilRefPic is derived by invoking the process specified in clause
H.8.1.4
            with picX and RefPicLayerId[ i ] given as inputs
      if( refPicSet0Flag ) {
   (H-1)
         RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = ilRefPic
         RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term
reference"
      } else {
         RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = ilRefPic
         RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term
reference"
      }
   } else {
      if( refPicSet0Flag )
         RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"
      else
         RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] = "no reference picture"
   }
}
```

There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1.

There shall be no picture that has discardable_flag equal to 1 in RefPicSetInterLayer0 or RefPicSetInterLayer1.

NOTE 1 – For the profiles defined in this annex, RefPicSetInterLayer1 is always empty since the value of ViewId[i] is equal to zero for all layers.

If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture.

NOTE 2 – An access unit may contain both RASL and RADL pictures.

H.8.1.4 Derivation process for inter-layer reference pictures

H.8.1.4.1 General

Inputs to this process are:

– a decoded direct reference layer picture rlPic,

– a variable rLId specifying the value of nuh_layer_id of the direct reference layer picture.

Output of this process is the inter-layer reference picture ilRefPic.

The variables PicWidthInSamplesCurrY, PicHeightInSamplesCurrY, BitDepthCurrY, BitDepthCurrC, SubWidthCurrC and SubHeightCurrC are set equal to pic_width_in_luma_samples, pic_height_in_luma_samples, $BitDepth_Y$, $BitDepth_C$, SubWidthC and SubHeightC of the current layer, respectively.

The variables PicWidthInSamplesRefLayerY and PicHeightInSamplesRefLayerY are set equal to the width and height of the decoded direct reference layer picture rlPic in units of luma samples, respectively. The variables BitDepthRefLayerY, BitDepthRefLayerC, SubWidthRefLayerC and SubHeightRefLayerC are set equal to the values of $BitDepth_Y$, $BitDepth_C$, SubWidthC and SubHeightC of the direct reference layer picture rlPic, respectively.

NOTE – This clause and its subclauses support all possible values of SubWidthCurrC, SubHeightCurrC, SubWidthRefLayerC and SubHeightRefLayerC. When the value of chroma_format_idc of the current layer and the value of chroma_format_idc of the direct reference layer are both equal to 1, the values of SubWidthCurrC, SubHeightCurrC, SubWidthRefLayerC and SubHeightRefLayerC are all equal to 2, and the resampling process of picture sample values in clause H.8.1.4.2 and the colour mapping process of picture sample values in clause H.8.1.4.4 can be simplified.

The variables RefLayerRegionLeftOffset, RefLayerRegionTopOffset, RefLayerRegionRightOffset and RefLayerRegionBottomOffset are derived as follows:

RefLayerRegionLeftOffset = ref_region_left_offset[rLId] * SubWidthRefLayerC (H-2)

RefLayerRegionTopOffset = ref_region_top_offset[rLId] * SubHeightRefLayerC (H-3)

RefLayerRegionRightOffset = ref_region_right_offset[rLId] * SubWidthRefLayerC (H-4)

RefLayerRegionBottomOffset = ref_region_bottom_offset[rLId] * SubHeightRefLayerC (H-5)

The variables RefLayerRegionWidthInSamplesY and RefLayerRegionHeightInSamplesY are the width and height of the reference region in the decoded direct reference layer picture rlPic in units of luma samples, respectively, and are derived as follows:

RefLayerRegionWidthInSamplesY = PicWidthInSamplesRefLayerY −
  RefLayerRegionLeftOffset − RefLayerRegionRightOffset (H-6)

RefLayerRegionHeightInSamplesY = PicHeightInSamplesRefLayerY −
  RefLayerRegionTopOffset − RefLayerRegionBottomOffset (H-7)

The variables PicWidthInSamplesCurrC, PicHeightInSamplesCurrC, PicWidthInSamplesRefLayerC and PicHeightInSamplesRefLayerC are derived as follows:

PicWidthInSamplesCurrC = PicWidthInSamplesCurrY / SubWidthCurrC (H-8)

PicHeightInSamplesCurrC = PicHeightInSamplesCurrY / SubHeightCurrC (H-9)

PicWidthInSamplesRefLayerC = PicWidthInSamplesRefLayerY / SubWidthRefLayerC (H-10)

PicHeightInSamplesRefLayerC = PicHeightInSamplesRefLayerY / SubHeightRefLayerC (H-11)

The variables ScaledRefLayerLeftOffset, ScaledRefLayerTopOffset, ScaledRefLayerRightOffset and ScaledRefLayerBottomOffset are derived as follows:

ScaledRefLayerLeftOffset = scaled_ref_layer_left_offset[rLId] * SubWidthCurrC (H-12)

ScaledRefLayerTopOffset = scaled_ref_layer_top_offset[rLId] * SubHeightCurrC (H-13)

ScaledRefLayerRightOffset = scaled_ref_layer_right_offset[rLId] * SubWidthCurrC (H-14)

ScaledRefLayerBottomOffset = scaled_ref_layer_bottom_offset[rLId] * SubHeightCurrC (H-15)

The variables ScaledRefRegionWidthInSamplesY and ScaledRefRegionHeightInSamplesY are derived as follows:

ScaledRefRegionWidthInSamplesY = PicWidthInSamplesCurrY − ScaledRefLayerLeftOffset − ScaledRefLayerRightOffset (H-16)

ScaledRefRegionHeightInSamplesY = PicHeightInSamplesCurrY − ScaledRefLayerTopOffset − ScaledRefLayerBottomOffset (H-17)

The variables SpatialScaleFactorHorY, SpatialScaleFactorVerY, SpatialScaleFactorHorC and SpatialScaleFactorVerC are derived as follows:

SpatialScaleFactorHorY = ((RefLayerRegionWidthInSamplesY << 16) + (ScaledRefRegionWidthInSamplesY >> 1)) / ScaledRefRegionWidthInSamplesY (H-18)

SpatialScaleFactorVerY = ((RefLayerRegionHeightInSamplesY << 16) + (ScaledRefRegionHeightInSamplesY >> 1)) / ScaledRefRegionHeightInSamplesY (H-19)

SpatialScaleFactorHorC = (((RefLayerRegionWidthInSamplesY / SubWidthRefLayerC) << 16) + ((ScaledRefRegionWidthInSamplesY / SubWidthCurrC) >> 1)) / (ScaledRefRegionWidthInSamplesY / SubWidthCurrC) (H-20)

SpatialScaleFactorVerC = (((RefLayerRegionHeightInSamplesY / SubHeightRefLayerC) << 16) + ((ScaledRefRegionHeightInSamplesY / SubHeightCurrC) >> 1)) / (ScaledRefRegionHeightInSamplesY / SubHeightCurrC) (H-21)

The variables PhaseHorY, PhaseVerY, PhaseHorC and PhaseVerC are set as follows:

PhaseHorY = phase_hor_luma[rLId] (H-22)

PhaseVerY = phase_ver_luma[rLId] (H-23)

PhaseHorC = phase_hor_chroma_plus8[rLId] − 8 (H-24)

PhaseVerC = phase_ver_chroma_plus8[rLId] − 8 (H-25)

It is a requirement of bitstream conformance that BitDepthRefLayerY shall be less than or equal to BitDepthCurrY and BitDepthRefLayerC shall be less than or equal to BitDepthCurrC.

It is a requirement of bitstream conformance that the values of RefLayerRegionWidthInSamplesY, RefLayerRegionHeightInSamplesY, ScaledRefRegionWidthInSamplesY and ScaledRefRegionHeightInSamplesY shall be greater than 0.

It is a requirement of bitstream conformance that ScaledRefRegionWidthInSamplesY shall be greater than or equal to RefLayerRegionWidthInSamplesY and ScaledRefRegionHeightInSamplesY shall be greater than or equal to RefLayerRegionHeightInSamplesY.

It is a requirement of bitstream conformance that, when ScaledRefRegionWidthInSamplesY is equal to RefLayerRegionWidthInSamplesY, PhaseHorY shall be equal to 0, when ScaledRefRegionWidthInSamplesC is equal to RefLayerRegionWidthInSamplesC, PhaseHorC shall be equal to 0, when ScaledRefRegionHeightInSamplesY is equal to RefLayerRegionHeightInSamplesY, PhaseVerY shall be equal to 0, and when ScaledRefRegionHeightInSamplesC is equal to RefLayerRegionHeightInSamplesC, PhaseVerC shall be equal to 0.

The inter-layer reference picture ilRefPic is derived as follows:

– The variables sampleProcessingFlag and motionProcessingFlag are initialized to 0.

– The variable equalPictureSizeAndOffsetFlag is derived as follows:

  – If all of the following conditions are true, equalPictureSizeAndOffsetFlag is set equal to 1:

- PicWidthInSamplesCurrY is equal to PicWidthInSamplesRefLayerY
- PicHeightInSamplesCurrY is equal to PicHeightInSamplesRefLayerY
- ScaledRefLayerLeftOffset is equal to RefLayerRegionLeftOffset
- ScaledRefLayerTopOffset is equal to RefLayerRegionTopOffset
- ScaledRefLayerRightOffset is equal to RefLayerRegionRightOffset
- ScaledRefLayerBottomOffset is equal to RefLayerRegionBottomOffset
- PhaseHorY, PhaseVerY, PhaseHorC and PhaseVerC are all equal to 0

- Otherwise, equalPictureSizeAndOffsetFlag is set equal to 0.

- The variable currColourMappingEnableFlag is derived as follows:
  - If colour_mapping_enabled_flag is equal to 1 and if there exists a value of i, with i in the range of 0 to num_cm_ref_layers_minus1, inclusive, for which cm_ref_layer_id[i] is equal to rLId, currColourMappingEnableFlag is set equal to 1.
  - Otherwise, currColourMappingEnableFlag is set equal to 0.
- If equalPictureSizeAndOffsetFlag is equal to 1, BitDepthRefLayerY is equal to BitDepthCurrY, BitDepthRefLayerC is equal to BitDepthCurrC, SubWidthRefLayerC is equal to SubWidthCurrC, SubHeightRefLayerC is equal to SubHeightCurrC and currColourMappingEnableFlag is equal to 0, ilRefPic is set equal to rlPic.
- Otherwise, the following applies:
  - The inter-layer reference picture ilRefPic is generated as follows:
    - The PicOrderCntVal value of ilRefPic is set equal to the PicOrderCntVal value of rlPic.
    - The nuh_layer_id value of ilRefPic is set equal to rLId.
    - The variable currLayerId is set equal to the value of nuh_layer_id of the current picture.
    - When VpsInterLayerSamplePredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLId]] is equal to 1, the following applies:
      - If currColourMappingEnableFlag is equal to 1, the following applies:
        - The colour mapping process as specified in clause H.8.1.4.4 is invoked with the picture sample arrays, $rlPicSample_L$, $rlPicSample_{Cb}$ and $rlPicSample_{Cr}$, of the direct reference layer picture rlPic as inputs, and with the colour mapped picture sample arrays, $cmPicSample_L$, $cmPicSample_{Cb}$ and $cmPicSample_{Cr}$ of the colour mapped reference layer picture cmPic as outputs.
        - BitDepthRefLayerY and BitDepthRefLayerC are set equal to BitDepthCmOutputY and BitDepthCmOutputC, respectively.
        - If equalPictureSizeAndOffsetFlag is equal to 1, BitDepthRefLayerY is equal to BitDepthCurrY, BitDepthRefLayerC is equal to BitDepthCurrC, SubWidthRefLayerC is equal to SubWidthCurrC and SubHeightRefLayerC is equal to SubHeightCurrC, the picture sample arrays $rsPicSample_L$, $rsPicSample_{Cb}$ and $rsPicSample_{Cr}$ of the inter-layer reference picture ilRefPic are set equal to $cmPicSample_L$, $cmPicSample_{Cb}$ and $cmPicSample_{Cr}$, respectively.
        - Otherwise, the picture sample resampling process as specified in clause H.8.1.4.2 is invoked with the picture sample arrays, $cmPicSample_L$, $cmPicSample_{Cb}$ and $cmPicSample_{Cr}$, of the colour mapped reference layer picture cmPic as inputs, and with the resampled picture sample arrays, $rsPicSample_L$, $rsPicSample_{Cb}$ and $rsPicSample_{Cr}$ of the inter-layer reference picture ilRefPic as outputs.
      - Otherwise, the picture sample resampling process as specified in clause H.8.1.4.2 is invoked with the picture sample arrays, $rlPicSample_L$, $rlPicSample_{Cb}$ and $rlPicSample_{Cr}$, of the direct reference layer picture rlPic as inputs, and with the resampled picture sample arrays, $rsPicSample_L$, $rsPicSample_{Cb}$ and $rsPicSample_{Cr}$ of the inter-layer reference picture ilRefPic as outputs.
      - sampleProcessingFlag is set equal to 1.
    - When VpsInterLayerMotionPredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLId]] is equal to 1, the following applies:
      - A single slice ilRefSlice of the inter-layer reference picture ilRefPic is generated as follows:

– The values of slice_type, num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1 for the generated slice ilRefSlice are set equal to the values of slice_type, num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1, respectively, of the first slice of rlPic.

– When ilRefSlice is a P or B slice, for i in the range of 0 to num_ref_idx_l0_active_minus1 of ilRefSlice, inclusive, the reference picture associated with index i in reference picture list 0 of ilRefSlice is set equal to the reference picture associated with index i in reference picture list 0 of the first slice of rlPic.

– When ilRefSlice is a B slice, for i in the range of 0 to num_ref_idx_l1_active_minus1 of ilRefSlice, inclusive, the reference picture associated with index i in reference picture list 1 of ilRefSlice is set equal to the reference picture associated with index i in reference picture list 1 of the first slice of rlPic.

NOTE – When the inter-layer reference picture ilRefPic is used as the collocated picture for temporal motion vector prediction, all slices of rlPic are constrained to have the same values of slice_type, num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1.

– If equalPictureSizeAndOffsetFlag is equal to 1, the following applies:

– The motion and mode parameters of the inter-layer reference picture ilRefPic, including an array CuPredMode specifying the prediction modes, two arrays RefIdxL0 and RefIdxL1 specifying the reference indices, two arrays MvL0 and MvL1 specifying the luma motion vectors, and two arrays PredFlagL0 and PredFlagL1 specifying the prediction list utilization flags, are set equal to those of the decoded direct reference layer picture rlPic, respectively.

– Otherwise, the following applies:

– The picture motion and mode parameters resampling process as specified in clause H.8.1.4.3 is invoked with the direct reference layer picture rlPic, an array rlPredMode specifying the prediction modes CuPredMode of rlPic, two arrays rlRefIdxL0 and rlRefIdxL1 specifying the reference indices of rlPic, two arrays rlMvL0 and rlMvL1 specifying the luma motion vectors of rlPic, and two arrays rlPredFlagL0 and rlPrefFlagL1 specifying the prediction list utilization flags of rlPic as inputs, and an array rsPredMode specifying the prediction modes CuPredMode of ilRefPic, two arrays rsRefIdxL0 and rsRefIdxL1 specifying the reference indices of ilRefPic, two arrays rsMvL0 and rsMvL1 specifying the luma motion vectors of ilRefPic, and two arrays rsPredFlagL0 and rsPredFlagL1 specifying the prediction list utilization flags of ilRefPic as outputs.

– motionProcessingFlag is set equal to 1.

– The following applies:

$$\text{NumRefLayerPicsSampleProcessing} \mathrel{+}= \text{sampleProcessingFlag} \quad \text{(H-26)}$$

$$\text{NumRefLayerPicsMotionProcessing} \mathrel{+}= \text{motionProcessingFlag} \quad \text{(H-27)}$$

$$\text{NumRefLayerPicsProcessing} \mathrel{+}= \text{sampleProcessingFlag} \mid\mid \text{motionProcessingFlag} \quad \text{(H-28)}$$

H.8.1.4.2 Resampling process of picture sample values

H.8.1.4.2.1 General

Inputs to this process are:

– a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array rlPicSample_L of luma samples,

– a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rlPicSample_{Cb} of chroma samples of the component Cb,

– a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rlPicSample_{Cr} of chroma samples of the component Cr.

Outputs of this process are:

– a (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) array rsPicSample_L of luma samples,

– a (PicWidthInSamplesCurrC)x(PicHeightInSamplesCurrC) array rsPicSample_{Cb} of chroma samples of the component Cb,

– a (PicWidthInSamplesCurrC)x(PicHeightInSamplesCurrC) array $rsPicSample_{Cr}$ of chroma samples of the component Cr.

The resampled luma sample value $rsPicSample_L$[xP][yP], with xP = 0..PicWidthInSamplesCurrY − 1, yP = 0..PicHeightInSamplesCurrY − 1, is derived by invoking the luma sample resampling process specified in clause H.8.1.4.2.2 with luma sample location (xP, yP) and the reference luma sample array $rlPicSample_L$ given as inputs.

The resampled chroma sample value $rsPicSample_{Cb}$[xP_C][yP_C], with xP_C = 0..PicWidthInSamplesCurrC − 1, yP_C = 0..PicHeightInSamplesCurrC − 1, of the chroma component Cb is derived by invoking the chroma sample resampling process specified in clause H.8.1.4.2.3 with chroma sample location (xP_C, yP_C) and the reference chroma sample array $rlPicSample_{Cb}$ given as inputs.

The resampled chroma sample value $rsPicSample_{Cr}$[xP_C][yP_C], with xP_C = 0..PicWidthInSamplesCurrC − 1, yP_C = 0..PicHeightInSamplesCurrC − 1, of the chroma component Cr is derived by invoking the chroma sample resampling process specified in clause H.8.1.4.2.3 with chroma sample location (xP_C, yP_C) and the reference sample array $rlPicSample_{Cr}$ given as inputs.

H.8.1.4.2.2 Resampling process of luma sample values

Inputs to this process are

– a luma sample location (xP, yP) relative to the top-left luma sample of the current picture,

– the luma reference sample array $rlPicSample_L$.

Output of this process is the resampled luma sample value rsLumaSample.

Table H.1 specifies the 8-tap filter coefficients f_L[p, x] with p = 0..15 and x = 0..7 used for the luma resampling process.

Table H.1 – 16-phase luma resampling filter

Phase p	Interpolation filter coefficients							
	f_L[p, 0]	f_L[p, 1]	f_L[p, 2]	f_L[p, 3]	f_L[p, 4]	f_L[p, 5]	f_L[p, 6]	f_L[p, 7]
0	0	0	0	64	0	0	0	0
1	0	1	−3	63	4	−2	1	0
2	−1	2	−5	62	8	−3	1	0
3	−1	3	−8	60	13	−4	1	0
4	−1	4	−10	58	17	−5	1	0
5	−1	4	−11	52	26	−8	3	−1
6	−1	3	−9	47	31	−10	4	−1
7	−1	4	−11	45	34	−10	4	−1
8	−1	4	−11	40	40	−11	4	−1
9	−1	4	−10	34	45	−11	4	−1
10	−1	4	−10	31	47	−9	3	−1
11	−1	3	−8	26	52	−11	4	−1
12	0	1	−5	17	58	−10	4	−1
13	0	1	−4	13	60	−8	3	−1
14	0	1	−3	8	62	−5	2	−1
15	0	1	−2	4	63	−3	1	0

The value of the resampled luma sample rsLumaSample is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location used in resampling as specified in clause H.8.1.4.2.4 is invoked with chromaFlag equal to 0 and luma sample location (xP, yP) given as the inputs and luma sample location (xRef16, yRef16) in units of 1/16-th luma sample as output.

2. The variables xRef and xPhase are derived as follows:

$$xRef = (xRef16 >> 4) \tag{H-29}$$

$$xPhase = (xRef16) \% 16 \tag{H-30}$$

3. The variables yRef and yPhase are derived as follows:

$$yRef = (yRef16 >> 4) \tag{H-31}$$

$$yPhase = (yRef16) \% 16 \tag{H-32}$$

4. The variables shift1, shift2 and offset are derived as follows:

$$shift1 = BitDepthRefLayerY - 8 \tag{H-33}$$

$$shift2 = 20 - BitDepthCurrY \tag{H-34}$$

$$offset = 1 << (shift2 - 1) \tag{H-35}$$

5. The sample value tempArray[n] with n = 0..7, is derived as follows:

$$yPosRL = Clip3(0, PicHeightInSamplesRefLayerY - 1, yRef + n - 3) \tag{H-36}$$

$$refW = PicWidthInSamplesRefLayerY \tag{H-37}$$

$$\begin{aligned} tempArray[n] = (& f_L[xPhase, 0] * rlPicSample_L[Clip3(0, refW - 1, xRef - 3), yPosRL] + \\ & f_L[xPhase, 1] * rlPicSample_L[Clip3(0, refW - 1, xRef - 2), yPosRL] + \\ & f_L[xPhase, 2] * rlPicSample_L[Clip3(0, refW - 1, xRef - 1), yPosRL] + \\ & f_L[xPhase, 3] * rlPicSample_L[Clip3(0, refW - 1, xRef), yPosRL] + \\ & f_L[xPhase, 4] * rlPicSample_L[Clip3(0, refW - 1, xRef + 1), yPosRL] + \\ & f_L[xPhase, 5] * rlPicSample_L[Clip3(0, refW - 1, xRef + 2), yPosRL] + \\ & f_L[xPhase, 6] * rlPicSample_L[Clip3(0, refW - 1, xRef + 3), yPosRL] + \\ & f_L[xPhase, 7] * rlPicSample_L[Clip3(0, refW - 1, xRef + 4), yPosRL]) >> shift1 \end{aligned} \tag{H-38}$$

6. The resampled luma sample value rsLumaSample is derived as follows:

$$\begin{aligned} rsLumaSample = (& f_L[yPhase, 0] * tempArray[0] + \\ & f_L[yPhase, 1] * tempArray[1] + \\ & f_L[yPhase, 2] * tempArray[2] + \\ & f_L[yPhase, 3] * tempArray[3] + \\ & f_L[yPhase, 4] * tempArray[4] + \\ & f_L[yPhase, 5] * tempArray[5] + \\ & f_L[yPhase, 6] * tempArray[6] + \\ & f_L[yPhase, 7] * tempArray[7] + offset) >> shift2 \end{aligned} \tag{H-39}$$

$$rsLumaSample = Clip3(0, (1 << BitDepthCurrY) - 1, rsLumaSample) \tag{H-40}$$

H.8.1.4.2.3 Resampling process of chroma sample values

Inputs to this process are:

– a chroma sample location (xP_C, yP_C) relative to the top-left chorma sample of the current picture,

– the chroma reference sample array $rlPicSample_C$.

Output of this process is the resampled chroma sample value rsChromaSample.

Table H.2 specifies the 4-tap filter coefficients f_C[p, x] with p = 0..15 and x = 0..3 used for the chroma resampling process.

Table H.2 – 16-phase chroma resampling filter

Phase p	Interpolation filter coefficients			
	f_C[p, 0]	f_C[p, 1]	f_C[p, 2]	f_C[p, 3]
0	0	64	0	0
1	−2	62	4	0
2	−2	58	10	−2
3	−4	56	14	−2
4	−4	54	16	−2
5	−6	52	20	−2
6	−6	46	28	−4
7	−4	42	30	−4

8	−4	36	36	−4
9	−4	30	42	−4
10	−4	28	46	−6
11	−2	20	52	−6
12	−2	16	54	−4
13	−2	14	56	−4
14	−2	10	58	−2
15	0	4	62	−2

The value of the resampled chroma sample value rsChromaSample is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location in resampling as specified in clause H.8.1.4.2.4 is invoked with chromaFlag equal to 1 and chroma sample location (xP_C, yP_C) given as the inputs and chroma sample location (xRef16, yRef16) in units of 1/16-th chroma sample as output.
2. The variables xRef and xPhase are derived as follows:

$$\text{xRef} = (\text{ xRef16 } >> 4) \quad \text{(H-41)}$$

$$\text{xPhase} = (\text{ xRef16 }) \,\%\, 16 \quad \text{(H-42)}$$

3. The variables yRef and yPhase are derived as follows:

$$\text{yRef} = (\text{ yRef16 } >> 4) \quad \text{(H-43)}$$

$$\text{yPhase} = (\text{ yRef16 }) \,\%\, 16 \quad \text{(H-44)}$$

4. The variables shift1, shift2 and offset are derived as follows:

$$\text{shift1} = \text{BitDepthRefLayerC} - 8 \quad \text{(H-45)}$$

$$\text{shift2} = 20 - \text{BitDepthCurrC} \quad \text{(H-46)}$$

$$\text{offset} = 1 << (\text{ shift2} - 1) \quad \text{(H-47)}$$

5. The sample value tempArray[n] with n = 0..3, is derived as follows:

$$\text{yPosRL} = \text{Clip3}(0, \text{PicHeightInSamplesRefLayerC} - 1, \text{yRef} + n - 1) \quad \text{(H-48)}$$

$$\text{refWC} = \text{PicWidthInSamplesRefLayerC} \quad \text{(H-49)}$$

$$\begin{aligned}\text{tempArray}[\,n\,] = (\, & f_C[\text{ xPhase}, 0] * \text{rlPicSample}_C[\text{ Clip3}(0, \text{refWC} - 1, \text{xRef} - 1), \text{yPosRL}] + \\ & f_C[\text{ xPhase}, 1] * \text{rlPicSample}_C[\text{ Clip3}(0, \text{refWC} - 1, \text{xRef}), \text{yPosRL}] + \\ & f_C[\text{ xPhase}, 2] * \text{rlPicSample}_C[\text{ Clip3}(0, \text{refWC} - 1, \text{xRef} + 1), \text{yPosRL}] + \quad \text{(H-50)} \\ & f_C[\text{ xPhase}, 3] * \text{rlPicSample}_C[\text{ Clip3}(0, \text{refWC} - 1, \text{xRef} + 2), \text{yPosRL}] \,) >> \text{shift1}\end{aligned}$$

6. The resampled chroma sample value rsChromaSample is derived as follows:

$$\begin{aligned}\text{rsChromaSample} = (\, & f_C[\text{ yPhase}, 0] * \text{tempArray}[0] + \\ & f_C[\text{ yPhase}, 1] * \text{tempArray}[1] + \\ & f_C[\text{ yPhase}, 2] * \text{tempArray}[2] + \quad \text{(H-51)} \\ & f_C[\text{ yPhase}, 3] * \text{tempArray}[3] + \text{offset} \,) >> \text{shift2}\end{aligned}$$

$$\text{rsChromaSample} = \text{Clip3}(0, (1 << \text{BitDepthCurrC}) - 1, \text{rsChromaSample}) \quad \text{(H-52)}$$

H.8.1.4.2.4 Derivation process for reference layer sample location in units of 1/16-th sample

Inputs to this process are:

– a variable chromaFlag specifying whether the sample location being derived is that of the luma component or that of one of the chroma colour components.

– a sample location (xP, yP) relative to the top-left sample of the luma component or the top-left sample of one of the chroma components of the current picture depending on the value of chromaFlag.

Output of this process is a sample location (xRef16, yRef16) specifying the reference layer sample location in units of 1/16-th sample relative to the top-left sample of the luma component or the top-left sample of one of the chroma components of the direct reference layer picture depending on the value of chromaFlag.

The variables currOffsetLeft, currOffsetTop, refOffsetLeft and refOffsetTop are derived as follows:

$$\text{currOffsetLeft} = \text{ScaledRefLayerLeftOffset} \,/\, (\ \text{chromaFlag} \ ?\ \text{SubWidthCurrC} : 1) \qquad \text{(H-53)}$$

$$\text{currOffsetTop} = \text{ScaledRefLayerTopOffset} \,/\, (\ \text{chromaFlag} \ ?\ \text{SubHeightCurrC} : 1) \qquad \text{(H-54)}$$

$$\text{refOffsetLeft} = (\ \text{RefLayerRegionLeftOffset} \,/\, (\ \text{chromaFlag} \ ?\ \text{SubWidthRefLayerC} : 1)\ ) \ll 4 \qquad \text{(H-55)}$$

$$\text{refOffsetTop} = (\ \text{RefLayerRegionTopOffset} \,/\, (\ \text{chromaFlag} \ ?\ \text{SubHeightRefLayerC} : 1)\ ) \ll 4 \qquad \text{(H-56)}$$

The variables phaseHor, phaseVer, scaleHor and scaleVer are derived as follows:

$$\text{phaseHor} = \text{chromaFlag} \ ?\ \text{PhaseHorC} : \text{PhaseHorY} \qquad \text{(H-57)}$$

$$\text{phaseVer} = \text{chromaFlag} \ ?\ \text{PhaseVerC} : \text{PhaseVerY} \qquad \text{(H-58)}$$

$$\text{scaleHor} = \text{chromaFlag} \ ?\ \text{SpatialScaleFactorHorC} : \text{SpatialScaleFactorHorY} \qquad \text{(H-59)}$$

$$\text{scaleVer} = \text{chromaFlag} ?\ \text{SpatialScaleFactorVerC} : \text{SpatialScaleFactorVerY} \qquad \text{(H-60)}$$

The variables addHor and addVer are derived as follows:

$$\text{addHor} = -(\ (\ \text{scaleHor} * \text{phaseHor} + 8\ ) \gg 4\ ) \qquad \text{(H-61)}$$

$$\text{addVer} = -(\ (\ \text{scaleVer} * \text{phaseVer} + 8\ ) \gg 4\ ) \qquad \text{(H-62)}$$

The variables xRef16 and yRef16 are derived as follows:

$$\text{xRef16} = (\ (\ (\ \text{xP} - \text{currOffsetLeft}\ ) * \text{scaleHor} + \text{addHor} + (\ 1 \ll 11\ )\ ) \gg 12\ ) + \text{refOffsetLeft} \qquad \text{(H-63)}$$

$$\text{yRef16} = (\ (\ (\ \text{yP} - \text{currOffsetTop}\ ) * \text{scaleVer} + \text{addVer} + (\ 1 \ll 11\ )\ ) \gg 12\ ) + \text{refOffsetTop} \qquad \text{(H-64)}$$

H.8.1.4.3 Resampling process of picture motion and mode parameters

Inputs to this process are:

- a decoded direct reference layer picture rlPic,
- a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array rlPredMode specifying the prediction modes of the direct reference layer picture,
- two (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) arrays rlRefIdxL0 and rlRefIdxL1 specifying the reference indices of the direct reference layer picture,
- two (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) arrays rlMvL0 and rlMvL1 specifying the luma motion vectors of the direct reference layer picture,
- two (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) arrays rlPredFlagL0 and rlPredFlagL1 specifying the prediction list utilization flags of the direct reference layer picture.

Outputs of this process are:

- a (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) array rsPredMode specifying the prediction modes of the resampled picture,

– two (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) arrays rsRefIdxL0 and rsRefIdxL1 specifying the reference indices of the resampled picture,

– two (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) arrays rsMvL0 and rsMvL1 specifying the luma motion vectors of the resampled picture,

– two (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) arrays rsPredFlagL0 and rsPredFlagL1 specifying the prediction list utilization flags of the resampled picture.

For each 16x16 prediction block of the resampled picture with the top-left luma sample location at (xP, yP), where xP = xB << 4 and yP = yB << 4, for xB = 0..((PicWidthInSamplesCurrY + 15) >> 4) − 1 and yB = 0..((PicHeightInSamplesCurrY + 15) >> 4) − 1, its motion and mode parameters rsPredMode[xP][yP], rsMvLX[xP][yP], rsRefIdxLX[xP][yP] and rsPredFlagLX[xP][yP], with X being equal to 0 and 1, are derived by applying the following ordered steps:

1. The centre location (xPCtr, yPCtr) of the luma prediction block is derived as follows:

xPCtr = xP + 8 (H-65)

yPCtr = yP + 8 (H-66)

2. The variables xRef and yRef are derived as follows:

xRef =
(((xPCtr − ScaledRefLayerLeftOffset) * SpatialScaleFactorHorY + (1 << 15)) >> 16)
+ RefLayerRegionLeftOffset
(H-67)

yRef =
(((yPCtr − ScaledRefLayerTopOffset) * SpatialScaleFactorVerY + (1 << 15)) >> 16)
+ RefLayerRegionTopOffset
(H-68)

3. The rounded reference layer luma sample location (xRL, yRL) is derived as follows:

xRL = ((xRef + 4) >> 4) << 4
(H-69)

yRL = ((yRef + 4) >> 4) << 4
(H-70)

4. The variable rsPredMode[xP][yP] is derived as follows:

```
if( xRL < 0 || xRL >= PicWidthInSamplesRefLayerY || yRL < 0 ||
      yRL >= PicHeightInSamplesRefLayerY ),                              (H-71)
   rsPredMode[ xP ][ yP ] = MODE_INTRA
else
   rsPredMode[ xP ][ yP ] = rlPredMode[ xRL ][ yRL ]
```

5. For X being each of 0 and 1, the variables rsMvLX[xP][yP], rsRefIdxLX[xP][yP] and rsPredFlagLX[xP][yP] are derived as follows:

   – If rsPredMode[xP][yP] is equal to MODE_INTER, the following applies:

      – rsRefIdxLX[xP][yP] and rsPredFlagLX[xP][yP] are derived as follows:

rsRefIdxLX[xP][yP] = rlRefIdxLX[xRL][yRL] (H-72)
rsPredFlagLX[xP][yP] = rlPredFlagLX[xRL][yRL] (H-73)

      – rsMvLX[xP][yP][0] is derived as follows:

– If ScaledRefRegionWidthInSamplesY is not equal to RefLayerRegionWidthInSamplesY, the following applies:

scaleMVX = Clip3(−4096, 4095, ((ScaledRefRegionWidthInSamplesY << 8) (H-74)
+ (RefLayerRegionWidthInSamplesY >> 1)) / RefLayerRegionWidthInSamplesY)

rsMvLX[xP][yP][0] = Clip3(−32768, 32767, Sign(scaleMVX *
rlMvLX[xRL][yRL][0]) * ((Abs (scaleMVX * rlMvLX[xRL][yRL][0])
(H-75)
+ 127) >> 8))

– Otherwise, the following applies:

rsMvLX[xP][yP][0] = rlMvLX[xRL][yRL][0] (H-76)

– rsMvLX[xP][yP][1] is derived as follows:

– If ScaledRefRegionHeightInSamplesY is not equal to RefLayerRegionHeightInSamplesY, the following applies:

scaleMVY = Clip3(−4096, 4095, ((ScaledRefRegionHeightInSamplesY << 8)(H-77)
+ (RefLayerRegionHeightInSamplesY >> 1)) /
RefLayerRegionHeightInSamplesY)

rsMvLX[xP][yP][1] = Clip3(−32768, 32767, Sign(scaleMVY *
rlMvLX[xRL][yRL][1]) * ((Abs (scaleMVY * rlMvLX[xRL][yRL][1])
(H-78)
+ 127) >> 8))

– Otherwise, the following applies:

rsMvLX[xP][yP][1] = rlMvLX[xRL][yRL][1] (H-79)

– Otherwise (rsPredMode[xP][yP] is equal to MODE_INTRA), the following applies:

– rsMvL0[xP][yP][0], rsMvL0[xP][yP][1], rsMvL1[xP][yP][0] and rsMvL1[xP][yP][1] are set equal to 0.

– rsRefIdxL0[xP][yP] and rsRefIdxL1[xP][yP] are set equal to −1.

– rsPredFlagL0[xP][yP] and rsPredFlagL1[xP][yP] are set equal to 0.

H.8.1.4.4 Colour mapping process of picture sample values

H.8.1.4.4.1 General

Inputs to this process are:

– a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array rlPicSample_{L} of luma samples,

– a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rlPicSample_{Cb} of chroma samples of the component Cb,

– a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rlPicSample_{Cr} of chroma samples of the component Cr.

Outputs of this process are:

– a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array cmPicSample_{L} of luma samples,

– a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array cmPicSample_{Cb} of chroma samples of the component Cb,

– a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array cmPicSample_{Cr} of chroma samples of the component Cr.

The colour mapped luma sample cmPicSampleL[xP][yP] with xP = 0..PicWidthInSamplesRefLayerY − 1, yP = 0..PicHeightInSamplesRefLayerY − 1 is derived by invoking the colour mapping process of luma sample values as

specified in clause H.8.1.4.4.2 with luma sample location (xP, yP), sample arrays rlPicSampleL, rlPicSampleCb and rlPicSampleCr given as inputs.

The colour mapped chroma sample cmPicSampleCb[xC][yC] with xC = 0..PicWidthInSamplesRefLayerC − 1, yC = 0..PicHeightInSamplesRefLayerC − 1 is derived by invoking the colour mapping process of chroma sample values as specified in clause H.8.1.4.4.3 with chroma sample location (xC, yC), sample arrays rlPicSampleL, rlPicSampleCb rlPicSampleCr and cIdx equal to 0 given as inputs.

The colour mapped chroma sample cmPicSampleCr[xC][yC] with xC = 0..PicWidthInSamplesRefLayerC − 1, yC = 0..PicHeightInSamplesRefLayerC − 1 is derived by invoking the colour mapping process of chroma sample values as specified in clause H.8.1.4.4.3 with chroma sample location (xC, yC), sample arrays rlPicSampleL, rlPicSampleCb, rlPicSampleCr and cIdx equal to 1 given as inputs.

H.8.1.4.4.2 Colour mapping process of luma sample values

Inputs to this process are:

– a luma sample location (xP, yP) specifying the luma sample location relative to the top-left luma sample of the direct reference layer picture,

– the luma reference layer sample array $rlPicSample_Y$

– the chroma reference layer sample array $rlPicSample_{Cb}$ of the Cb component

– the chroma reference layer sample array $rlPicSample_{Cr}$ of the Cr component

Output of the process is a colour mapped luma sample value cmLumaSample

The chroma sample location (xP_C, yP_C) is set equal to (xP / SubWidthRefLayerC, yP / SubHeightRefLayerC).

The value of cmLumaSample is derived by applying the following ordered steps:

1. The variables yShift2Idx, cShift2Idx are derived as follows:

$$yShift2Idx = BitDepthCmInputY - cm\_octant\_depth - cm\_y\_part\_num\_log2 \quad (H\text{-}80)$$

$$cShift2Idx = BitDepthCmInputC - cm\_octant\_depth \quad (H\text{-}81)$$

2. The variables nMappingShift and nMappingOffset are derived as follows:

$$nMappingShift = 10 + BitDepthCmInputY - BitDepthCmOutputY \quad (H\text{-}82)$$

$$nMappingOffset = 1 \ll (nMappingShift - 1) \quad (H\text{-}83)$$

3. The variables tempCb and tempCr are derived as follows:

   – If SubWidthRefLayerC is equal to 2 and SubHeightRefLayerC is equal to 2, the following applies:

     – If xP % 2 is equal to 0 and yP % 2 is equal to 0, the following applies:

$$yP2_C = Max(0, yP_C - 1) \quad (H\text{-}84)$$

$$tempCb = (rlPicSample_{Cb}[xP_C][yP_C] * 3 + rlPicSample_{Cb}[xP_C][yP2_C] + 2) \gg 2 \quad (H\text{-}85)$$

$$tempCr = (rlPicSample_{Cr}[xP_C][yP_C] * 3 + rlPicSample_{Cr}[xP_C][yP2_C] + 2) \gg 2 \quad (H\text{-}86)$$

     – Otherwise, if xP % 2 is equal to 0 and yP % 2 is equal to 1, the following applies:

$$yP2_C = Min(yP_C + 1, PicHeightInSamplesRefLayerC - 1) \quad (H\text{-}87)$$

$$tempCb = (rlPicSample_{Cb}[xP_C][yP_C] * 3 + rlPicSample_{Cb}[xP_C][yP2_C] + 2) \gg 2 \quad (H\text{-}88)$$

$$tempCr = (rlPicSample_{Cr}[xP_C][yP_C] * 3 + rlPicSample_{Cr}[xP_C][yP2_C] + 2) \gg 2 \quad (H\text{-}89)$$

– Otherwise, if xP % 2 is equal to 1 and yP % 2 is equal to 0, the following applies:

$$xP2_C = \text{Min}(\ xP_C + 1, \text{PicWidthInSamplesRefLayerC} - 1\ ) \quad \text{(H-90)}$$

$$yP2_C = \text{Max}(\ 0, yP_C - 1\ ) \quad \text{(H-91)}$$

$$\text{tempCb} = (\ \text{rlPicSampleCb}[\ xP_C\ ][\ yP2_C\ ] + \text{rlPicSampleCb}[\ xP2_C\ ][\ yP2_C\ ] + (\ \text{rlPicSampleCb}[\ xP_C\ ][\ yP_C\ ] + \text{rlPicSampleCb}[\ xP2_C\ ][\ yP_C\ ]\ ) * 3 + 4\ ) >> 3 \quad \text{(H-92)}$$

$$\text{tempCr} = (\ \text{rlPicSampleCr}[\ xP_C\ ][\ yP2_C\ ] + \text{rlPicSampleCr}[\ xP2_C\ ][\ yP2_C\ ] + (\ \text{rlPicSampleCr}[\ xP_C\ ][\ yP_C\ ] + \text{rlPicSampleCr}[\ xP2_C\ ][\ yP_C\ ]\ ) * 3 + 4\ ) >> 3 \quad \text{(H-93)}$$

– Otherwise (xP % 2 is equal to 1 and yP % 2 is equal to 1), the following applies:

$$xP2_C = \text{Min}(\ xP_C + 1, \text{PicWidthInSamplesRefLayerC} - 1\ ) \quad \text{(H-94)}$$

$$yP2_C = \text{Min}(\ yP_C + 1, \text{PicHeightInSamplesRefLayerC} - 1\ ) \quad \text{(H-95)}$$

$$\text{tempCb} = (\ (\ \text{rlPicSampleCb}[\ xP_C\ ][\ yP_C\ ] + \text{rlPicSampleCb}[\ xP2_C\ ][\ yP_C\ ]\ ) * 3 + \text{rlPicSampleCb}[\ xP_C\ ][\ yP2_C\ ] + \text{rlPicSampleCb}[\ xP2_C\ ][\ yP2_C\ ] + 4\ ) >> 3 \quad \text{(H-96)}$$

$$\text{tempCr} = (\ (\ \text{rlPicSampleCr}[\ xPC\ ][\ yPC\ ] + \text{rlPicSampleCr}[\ xP2_C\ ][\ yP_C\ ]\ ) * 3 + \text{rlPicSampleCr}[\ xP_C\ ][\ yP2_C\ ] + \text{rlPicSampleCr}[\ xP2_C\ ][\ yP2_C\ ] + 4\ ) >> 3 \quad \text{(H-97)}$$

– Otherwise, if SubWidthRefLayerC is equal to 2, the following applies:

– If xP % 2 is equal to 1, the following applies:

$$xP2_C = \text{Min}(\ xP_C + 1, \text{PicWidthInSamplesRefLayerC} - 1\ ) \quad \text{(H-98)}$$

$$\text{tempCb} = (\ \text{rlPicSampleCb}[\ xP_C\ ][\ yP_C\ ] + \text{rlPicSampleCb}[\ xP2_C\ ][\ yP_C\ ] + 1\ ) >> 1 \quad \text{(H-99)}$$

$$\text{tempCr} = (\ \text{rlPicSampleCr}[\ xP_C\ ][\ yP_C\ ] + \text{rlPicSampleCr}[\ xP2_C\ ][\ yP_C\ ] + 1\ ) >> 1 \quad \text{(H-100)}$$

– Otherwise (xP % 2 is equal to 0), the following applies:

$$\text{tempCb} = \text{rlPicSampleCb}[\ xP_C\ ][\ yP_C\ ] \quad \text{(H-101)}$$

$$\text{tempCr} = \text{rlPicSampleCr}[\ xP_C\ ][\ yP_C\ ] \quad \text{(H-102)}$$

– Otherwise (SubWidthRefLayerC is equal to 1 and SubHeightRefLayerC is equal to 1), the following applies:

$$\text{tempCb} = \text{rlPicSampleCb}[\ xP_C\ ][\ yP_C\ ] \quad \text{(H-103)}$$

$$\text{tempCr} = \text{rlPicSampleCr}[\ xP_C\ ][\ yP_C\ ] \quad \text{(H-104)}$$

4. The value of cmLumaSample is derived as follows:

$$\text{idxY} = \text{rlPicSample}_Y[\ xP\ ][\ yP\ ] >> \text{yShift2Idx} \quad \text{(H-105)}$$

$$\text{idxCb} = (\ \text{cm\_octant\_depth} == 1\ )\ ?\ (\ \text{tempCb} >= \text{CMThreshU}\ ) : (\ \text{tempCb} >> \text{cShift2Idx}\ ) \quad \text{(H-106)}$$

$$\text{idxCr} = (\ \text{cm\_octant\_depth} == 1\ )\ ?\ (\ \text{tempCr} >= \text{CMThreshV}\ ) : (\ \text{tempCr} >> \text{cShift2Idx}\ ) \quad \text{(H-107)}$$

$$cmLumaSample = ((LutY[idxY][idxCb][idxCr][0] * rlPicSample_Y[xP][yP] + LutY[idxY][idxCb][idxCr][1] * tempCb + LutY[idxY][idxCb][idxCr][2] * tempCr + nMappingOffset) >> nMappingShift) + LutY[idxY][idxCb][idxCr][3] \quad \text{(H-108)}$$

$$cmLumaSample = Clip3(0, (1 << BitDepthCmOutputY) - 1, cmLumaSample) \quad \text{(H-109)}$$

H.8.1.4.4.3 Colour mapping process of chroma sample values

Inputs to this process are:

– a chroma sample location (xP_C, yP_C) specifying the chroma sample location relative to the top-left chroma sample of the direct reference layer picture,

– the luma reference sample array $rlPicSample_Y$,

– the chroma reference sample array $rlPicSample_{Cb}$ of the Cb component,

– the chroma reference sample array $rlPicSample_{Cr}$ of the Cr component,

– a variable cIdx specifying the chroma component index.

Output of the process is a colour mapped chorma sample value cmChromaSample.

The luma sample location (xP, yP) is set equal to (xP_C * SubWidthRefLayerC, yP_C * SubHeightRefLayerC).

The colour mapping table LutC is set equal to LutCb if cIdx is equal to 0 and set equal to LutCr otherwise.

The value of cmChromaSample is derived by applying the following ordered steps:

1. The variables yShift2Idx, cShift2Idx are derived as follows:

$$yShift2Idx = BitDepthCmInputY - cm\_octant\_depth - cm\_y\_part\_num\_log2 \quad \text{(H-110)}$$

$$cShift2Idx = BitDepthCmInputC - cm\_octant\_depth \quad \text{(H-111)}$$

2. The variables nMappingShift and nMappingOffset are derived as follows:

$$nMappingShift = 10 + BitDepthCmInputY - BitDepthCmOutputY \quad \text{(H-112)}$$

$$nMappingOffset = 1 << (nMappingShift - 1) \quad \text{(H-113)}$$

3. The variable tempY is derived as follows:

   – If SubWidthRefLayerC is equal to 2 and SubHeightRefLayerC is equal to 2, the following applies:

$$tempY = (rlPicSample_Y[xP][yP] + rlPicSample_Y[xP][yP + 1] + 1) >> 1 \quad \text{(H-114)}$$

   – Otherwise (SubWidthRefLayerC is not equal to 2 or SubHeightRefLayerC is not equal to 2), the following applies:

$$tempY = rlPicSample_Y[xP][yP] \quad \text{(H-115)}$$

4. The value of cmChromaSample is derived as follows:

$$idxY = tempY >> yShift2Idx \quad \text{(H-116)}$$

$$idxCb = (cm\_octant\_depth == 1) ? (rlPicSample_{Cb}[xP_C][yP_C] >= CMThreshU) : (rlPicSample_{Cb}[xP_C][yP_C] >> cShift2Idx) \quad \text{(H-117)}$$

$$idxCr = (cm\_octant\_depth == 1) ? (rlPicSample_{Cr}[xP_C][yP_C] >= CMThreshV) : (rlPicSample_{Cr}[xP_C][yP_C] >> cShift2Idx) \quad \text{(H-118)}$$

$$\begin{aligned}\text{cmChromaSample} = ((\text{LutC}[\ \text{idxY}\ ][\ \text{idxCb}\ ][\ \text{idxCr}\ ][\ 0\ ] * \text{tempY} \\ + \text{LutC}[\ \text{idxY}\ ][\ \text{idxCb}\ ][\ \text{idxCr}\ ][\ 1\ ] * \text{rlPicSample}_{\text{Cb}}[\ \text{xP}_{\text{C}}\ ][\ \text{yP}_{\text{C}}\ ] \\ + \text{LutC}[\ \text{idxY}\ ][\ \text{idxCb}\ ][\ \text{idxCr}\ ][\ 2\ ] * \text{rlPicSample}_{\text{Cr}}[\ \text{xP}_{\text{C}}\ ][\ \text{yP}_{\text{C}}\ ] \\ + \text{nMappingOffset}) >> \text{nMappingShift}) + \text{LutC}[\ \text{idxY}\ ][\ \text{idxCb}\ ][\ \text{idxCr}\ ][\ 3\ ]\end{aligned} \quad \text{(H-119)}$$

$$\text{cmChromaSample} = \text{Clip3}(0, (1 << \text{BitDepthCmOutputC}) - 1, \text{cmChromaSample}) \quad \text{(H-120)}$$

H.8.2 NAL unit decoding process

The specification in clause F.8.2 apply.

H.8.3 Slice decoding processes

H.8.3.1 Decoding process for picture order count

The specifications in clause F.8.3.1 apply.

H.8.3.2 Decoding process for reference picture set

The specifications in clause F.8.3.2 apply.

H.8.3.3 Decoding process for generating unavailable reference pictures

The specifications in clause F.8.3.3 apply.

H.8.3.4 Decoding process for reference picture lists construction

The specifications in clause F.8.3.4 apply with the following additions:

NOTE – Because bitstreams conforming to this annex are constrained to allow only zero-valued motion vectors for inter prediction using inter-layer reference pictures, it is suggested that a scalable encoder should disable temporal motion vector prediction for the current picture (by setting slice_temporal_mvp_enabled_flag to zero) when the reference picture lists of all slices in the current picture include only inter-layer reference pictures. This way, the encoder would be able to avoid the need to send the slice segment header syntax elements collocated_from_l0_flag and collocated_ref_idx.

H.8.4 Decoding process for coding units coded in intra prediction mode

The specifications in clause F.8.4 apply.

H.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in clause F.8.5 apply with the following additions:

It is a requirement of bitstream conformance that, for X being replaced by either 0 or 1, the variables mvLX[0] and mvLX[1] as an output of clause 8.5.3.1 shall be equal to 0 if the value of refIdxLX as an output of clause 8.5.3.1 corresponds to an inter-layer reference picture. That is, in any conforming bitstream, for X being replaced by either 0 or 1, upon invoking the decoding process in clause 8.5.3.1, the values of the syntax elements merge_idx, mvp_lX_flag, ref_idx_lX, MvdLX and mvd_l1_zero_flag shall always result in zero values for mvLX[0] and mvLX[1] when the value of refIdxLX of the reference picture list RefPicListX indicates an inter-layer reference picture.

The variable currLayerId is set equal to nuh_layer_id of the current picture.

It is a requirement of bitstream conformance that when the reference picture represented by the variable refIdxLX and derived by invoking clause 8.5.3.2, for X being replaced by either 0 or 1, is an inter-layer reference picture, VpsInterLayerSamplePredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLId]] shall be equal to 1, where rLId is set equal to nuh_layer_id of the direct reference layer picture from which the inter-layer reference picture is derived.

It is a requirement of bitstream conformance that when the collocated picture colPic, used for temporal motion vector prediction and derived by invoking clause 8.5.3.2.7, is an inter-layer reference picture, VpsInterLayerMotionPredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLId]] shall be equal to 1, where rLId is set equal to nuh_layer_id of the direct reference layer picture from which the inter-layer reference picture is derived.

It is a requirement of bitstream conformance that the collocated picture colPic, used for temporal motion vector prediction and derived by invoking clause 8.5.3.2.7, shall not be an inter-layer reference picture if the direct reference layer picture, from which the inter-layer reference picture is derived, is coded using two or more slice segments and any of the following conditions is true:

– The slice segment header syntax element slice_type of at least one of the slice segments of the direct reference layer picture is different from the slice segment header syntax element slice_type of another slice segment of the direct reference layer picture;
– For X being replaced by either 0 or 1, the slice segment header syntax element num_ref_idx_lX_active_minus1 of at least one of the slice segments of the direct reference layer picture is different from the slice segment header syntax element num_ref_idx_lX_active_minus1 of another slice segment of the direct reference layer picture;
– For X being replaced by either 0 or 1, the reference picture list RefPicListX of at least one of the slice segments of the direct reference layer picture is different from the reference picture list RefPicListX of another slice segment of the direct reference layer picture.

H.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause F.8.6 apply.

H.8.7 In-loop filter process

The specifications in clause F.8.7 apply.

H.9 Parsing process

The specifications in clause F.9 apply.

H.10 Specification of bitstream subsets

The specifications in clause F.10 and its subclauses apply.

H.11 Profiles, tiers and levels

H.11.1 Profiles

H.11.1.1 Scalable Main and Scalable Main 10 profiles

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the Scalable Main or Scalable Main 10 profile, the following applies:

– Let olsIdx be the OLS index of the OLS, the sub-bitstream subBitstream and the base layer sub-bitstream baseBitstream are derived as specified in clause F.11.3.

When vps_base_layer_internal_flag is equal to 1, the base layer sub-bitstream baseBitstream shall obey the following constraints:

– When the layer conforms to the Scalable Main profile, the base layer sub-bitstream baseBitstream shall be indicated to conform to the Main profile.

– When the layer conforms to the Scalable Main 10 profile, the base layer sub-bitstream baseBitstream shall be indicated to conform to the Main 10 profile or the Main profile.

The sub-bitstream subBitstream shall obey the following constraints:

– All active VPSs shall have vps_num_rep_formats_minus1 in the range of 0 to 15, inclusive.

– All active SPSs for layers in subBitstream shall have chroma_format_idc equal to 1 only.

– All active SPSs for layers in subBitstream shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag, extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag and cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.

– CtbLog2SizeY derived from all active SPSs for layers in subBitstream shall be in the range of 4 to 6, inclusive.

– All active PPSs for layers in subBitstream shall have log2_max_transform_skip_block_size_minus2 and chroma_qp_offset_list_enabled_flag, when present, equal to 0 only.

– The variables NumRefLayerPicsProcessing, NumRefLayerPicsSampleProcessing and NumRefLayerPicsMotionProcessing shall be less than or equal to 1 for each decoded picture with nuh_layer_id included in layerIdListTarget that was used to derive subBitstream.

– ScalabilityId[j][smIdx] derived according to any active VPS shall be equal to 0 for any smIdx value not equal to 2 or 3 and for any value of j such that layer_id_in_nuh[j] is among layerIdListTarget that was used to derive subBitstream.

– For a layer with nuh_layer_id iNuhLId equal to any value included in layerIdListTarget that was used to derive subBitstream, the value of NumRefLayers[iNuhLId], which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 4.

– All active SPSs for layers in subBitstream shall have sps_range_extension_flag and sps_scc_extension_flag equal to 0 only.

– All active PPSs for layers in subBitstream shall have pps_range_extension_flag and pps_scc_extension_flag equal to 0 only.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for any layer in subBitstream shall not be equal to 255 (which indicates level 8.5).

– For any active VPS, DependencyId[i] shall be greater than DependencyId[j] for any values of i and j among layerIdListTarget that was used to derive subBitstream such that AuxId[i] is equal to AuxId[j] and i is greater than j.

When the layer conforms to the Scalable Main profile, the sub-bitstream subBitstream shall obey the following constraints:

– All active SPSs for layers in subBitstream shall have bit_depth_luma_minus8 equal to 0 only.

– All active SPSs for layers in subBitstream shall have bit_depth_chroma_minus8 equal to 0 only.

– All active PPSs for layers in subBitstream shall have colour_mapping_enabled_flag equal to 0 only.

– The tier and level constraints specified for the Scalable Main profile in clause H.11.2 shall be fulfilled.

When the layer conforms to the Scalable Main 10 profile, the sub-bitstream subBitstream shall obey the following constraints:

– All active SPSs for layers in subBitstream shall have bit_depth_luma_minus8 in the range of 0 to 2, inclusive.

– All active SPSs for layers in subBitstream shall have bit_depth_chroma_minus8 in the range of 0 to 2, inclusive.

– The tier and level constraints specified for the Scalable Main profile in clause H.11.2 shall be fulfilled.

In the remainder of this clause and clause H.11.2.1, all syntax elements in the profile_tier_level( ) syntax structure refer to those in the profile_tier_level( ) syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the Scalable Main profile is indicated as follows:

– If OpTid of the output operation point is equal to vps_max_sub_layer_minus1, the conformance is indicated by general_profile_idc being equal to 7 or general_profile_compatibility_flag[7] being equal to 1, and general_max_12bit_constraint_flag being equal to 1, general_max_10bit_constraint_flag being equal to 1, general_max_8bit_constraint_flag being equal to 1, general_max_422chroma_constraint_flag being equal to 1, general_max_420chroma_constraint_flag being equal to 1, general_max_monochrome_constraint_flag being equal to 0, general_intra_constraint_flag being equal to 0, general_one_picture_only_constraint_flag being equal to 0 and general_lower_bit_rate_constraint_flag being equal to 1.

– Otherwise (OpTid of the output operation point is less than vps_max_sub_layer_minus1), the conformance is indicated by sub_layer_profile_idc[OpTid] being equal to 7 or sub_layer_profile_compatibility_flag[OpTid][7] being equal to 1, and sub_layer_max_12bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_10bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_8bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_422chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_420chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_monochrome_constraint_flag[OpTid] being equal to 0, sub_layer_intra_constraint_flag[OpTid] being equal to 0, and sub_layer_one_picture_only_constraint_flag[OpTid] being equal to 0 and sub_layer_lower_bit_rate_constraint_flag[OpTid] being equal to 1.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the Scalable Main 10 profile is indicated as follows:

– If OpTid of the output operation point is equal to vps_max_sub_layer_minus1, the conformance is indicated by general_profile_idc being equal to 7 or general_profile_compatibility_flag[7] being equal to 1, and general_max_12bit_constraint_flag being equal to 1, general_max_10bit_constraint_flag being equal to 1, general_max_8bit_constraint_flag being equal to 0, general_max_422chroma_constraint_flag being equal to 1, general_max_420chroma_constraint_flag being equal to 1, general_max_monochrome_constraint_flag being equal to 0, general_intra_constraint_flag being equal to 0, general_one_picture_only_constraint_flag being equal to 0 and general_lower_bit_rate_constraint_flag being equal to 1.

– Otherwise (OpTid of the output operation point is less than vps_max_sub_layer_minus1), the conformance is indicated by sub_layer_profile_idc[OpTid] being equal to 7 or sub_layer_profile_compatibility_flag[OpTid][7] being equal to 1, and sub_layer_max_12bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_10bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_8bit_constraint_flag[OpTid] being equal to 0, sub_layer_max_422chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_420chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_monochrome_constraint_flag[OpTid] being equal to 0, sub_layer_intra_constraint_flag[OpTid] being equal to 0, and sub_layer_one_picture_only_constraint_flag[OpTid] being equal to 0 and sub_layer_lower_bit_rate_constraint_flag[OpTid] being equal to 1.

H.11.1.2 Scalable format range extensions profiles

The following profiles, collectively referred to as the scalable format range extensions profiles, are specified in this clause:

– The Scalable Monochrome, Scalable Monochrome 12 and Scalable Monochrome 16 profiles

– The Scalable Main 4:4:4 profile

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the Scalable Monochrome, Scalable Monochrome 12, Scalable Monochrome 16 or Scalable Main 4:4:4 profile, the following applies:

– Let olsIdx be the OLS index of the OLS, the sub-bitstream subBitstream and the base layer sub-bitstream baseBitstream are derived as specified in clause F.11.3.

When vps_base_layer_internal_flag is equal to 1, the base layer sub-bitstream baseBitstream shall obey the following constraints:

– The base layer sub-bitstream baseBitstream shall be indicated to conform to the Main profile, the Main 10 profile or a format range extensions profile.

The sub-bitstream subBitstream shall obey the following constraints:

– All active VPSs shall have vps_num_rep_formats_minus1 in the range of 0 to 15, inclusive.

– All active SPSs for layers in subBitstream shall have separate_colour_plane_flag, cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.

– CtbLog2SizeY derived from all active SPSs for layers in subBitstream shall be in the range of 4 to 6, inclusive.

– The variables NumRefLayerPicsProcessing, NumRefLayerPicsSampleProcessing, and NumRefLayerPicsMotionProcessing shall be less than or equal to 1 for each decoded picture with nuh_layer_id included in layerIdListTarget that was used to derive subBitstream.

– ScalabilityId[j][smIdx] derived according to any active VPS shall be equal to 0 for any smIdx value not equal to 2 or 3 and for any value of j such that layer_id_in_nuh[j] is among layerIdListTarget that was used to derive subBitstream.

– For a layer with nuh_layer_id iNuhLId equal to any value included in layerIdListTarget that was used to derive subBitstream, the value of NumRefLayers[iNuhLId], which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 4.

– The constraints specified in Table H.3, in which entries marked with "–" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element, shall apply for all active SPSs and PPSs for layers in subBitstream.

  NOTE – For some syntax elements with table entries marked with "–", a constraint may be imposed indirectly – e.g., by semantics constraints that are imposed elsewhere in this Specification when other specified constraints are fulfilled.

– All active SPSs for layers in subBitstream shall have the same value of chroma_format_idc.

– All active SPSs for layers in subBitstream shall have sps_scc_extension_flag equal to 0 only.

– All active PPSs for layers in subBitstream shall have pps_scc_extension_flag equal to 0 only.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– For any active VPS, DependencyId[i] shall be greater than DependencyId[j] for any values of i and j among layerIdListTarget that was used to derive subBitstream such that AuxId[i] is equal to AuxId[j] and i is greater than j.

– In bitstreams conforming to the Scalable Monochrome, Scalable Monochrome 12, Scalable Monochrome 16 or Scalable Main 4:4:4 profiles, general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for all layers shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the Scalable Monochrome, Scalable Monochrome 12, Scalable Monochrome 16, or Scalable Main 4:4:4 profiles in clause H.11.2, as applicable, shall be fulfilled.

Table H.3 – Allowed values for syntax elements in the scalable format range extensions profiles

Profile for which constraint is specified	chroma_format_idc	bit_depth_luma_minus8 and bit_depth_chroma_minus8	transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag, intra_smoothing_disabled_flag, persistent_rice_adaptation_enabled_flag, and log2_max_transform_skip_block_size_minus2	extended_precision_processing_flag	chroma_qp_offset_list_enabled_flag
Scalable Monochrome	0	0	0	0	0
Scalable Monochrome 12	0	0..4	0	0	0
Scalable Monochrome 16	0	–	–	–	0
Scalable Main 4:4:4	–	0	–	0	–

In the remainder of this clause and clause H.11.2.1, all syntax elements in the profile_tier_level( ) syntax structure refer to those in the profile_tier_level( ) syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream for the scalable format range exensions profiles is indicated as follows:

– If OpTid of the output operation point is equal to vps_max_sub_layer_minus1, the conformance is indicated by general_profile_idc being equal to 10 or general_profile_compatibility_flag[10] being equal to 1, with the additional indications specified in Table H.4 for the general constraint flags.

– Otherwise (OpTid of the output operation point is less than vps_max_sub_layer_minus1), the conformance is indicated by sub_layer_profile_idc[OpTid] being equal to 10 or sub_layer_profile_compatibility_flag[OpTid][10] being equal to 1, with the additional indications specified in Table H.4 for the flags associated with the index OpTid.

All other combinations of general_max_14bit_constraint_flag, general_max_12bit_constraint_flag, general_max_10bit_constraint_flag, general_max_8bit_constraint_flag, general_max_422chroma_constraint_flag, general_max_420chroma_constraint_flag, general_max_monochrome_constraint_flag, general_intra_constraint_flag, general_one_picture_only_constraint_flag, and general_lower_bit_rate_constraint_flag with general_profile_idc equal to 10 or

general_profile_compatibility_flag[10] equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of sub_layer_max_14bit_constraint_flag[OpTid], sub_layer_max_12bit_constraint_flag[OpTid], sub_layer_max_10bit_constraint_flag[OpTid], sub_layer_max_8bit_constraint_flag[OpTid], sub_layer_max_422chroma_constraint_flag[OpTid], sub_layer_max_420chroma_constraint_flag[OpTid], sub_layer_max_monochrome_constraint_flag[OpTid], sub_layer_intra_constraint_flag[OpTid], sub_layer_one_picture_only_constraint_flag[OpTid], and sub_layer_lower_bit_rate_constraint_flag[OpTid] with sub_layer_profile_idc[OpTid] equal to 10 or sub_layer_profile_compatibility_flag[OpTid][10] equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the scalable format range extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table H.4 – Bitstream indications for conformance to scalable range extensions profiles

Profile for which the bitstream indicates conformance	general_max_14bit_constraint_flag or sub_layer_max_14bit_constraint_flag[OpTid]	general_max_12bit_constraint_flag or sub_layer_max_12bit_constraint_flag[OpTid]	general_max_10bit_constraint_flag or sub_layer_max_10bit_constraint_flag[OpTid]	general_max_8bit_constraint_flag or sub_layer_max_8bit_constraint_flag[OpTid]	general_max_422chroma_constraint_flag or sub_layer_max_422chroma_constraint_flag[OpTid]	general_max_420chroma_constraint_flag or sub_layer_max_420chroma_constraint_flag[OpTid]	general_max_monochrome_constraint_flag or sub_layer_max_monochrome_constraint_flag[OpTid]	general_intra_constraint_flag or sub_layer_intra_constraint_flag[OpTid]	general_one_picture_only_constraint_flag or sub_layer_one_picture_only_constraint_flag[OpTid]	general_lower_bit_rate_constraint_flag or sub_layer_lower_bit_rate_constraint_flag[OpTid]
Scalable Monochrome	1	1	1	1	1	1	1	0	0	1
Scalable Monochrome 12	1	1	0	0	1	1	1	0	0	1
Scalable Monochrome 16	0	0	0	0	1	1	1	0	0	1
Scalable Main 4:4:4	1	1	1	1	0	0	0	0	0	1

H.11.2 Tiers and levels

H.11.2.1 General tier and level limits

For purposes of comparison of tier capabilities, the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 0 is considered to be a lower tier than the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 1.

For purposes of comparison of level capabilities, a particular level of a specific tier is considered to be a lower level than some other level of the same tier when the value of the general_level_idc or sub_layer_level_idc[i] of the particular level is less than that of the other level.

The following is specified for expressing the constraints in this clause and clause H.11.2.2:

– For the Scalable Main profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor and MinCrScaleFactor is the same as that specified in Table A.10 for the Main profile. For the Scalable Main 10 profile, the value of each of these variables is the same as that specified in Table A.10 for the Main 10 profile.

– For the Scalable Monochrome profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Monochrome profile.

– For the Scalable Monochrome 12 profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Monochrome 12 profile.

– For the Scalable Monochrome 16 profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Monochrome 16 profile.

– For the Scalable Main 4:4:4 profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Main 4:4:4 profile.

– Let access unit n be the n-th access unit in decoding order, with the first access unit being access unit 0 (i.e., the 0-th access unit).

– Let the variable fR be set equal to 1 ÷ 300.

– Let the variable olsIdx be the index of the OLS.

– For each layer with nuh_layer_id equal to currLayerId, let the variable layerSizeInSamplesY be derived as follows:

$$\text{layerSizeInSamplesY} = \text{pic\_width\_vps\_in\_luma\_samples} * \text{pic\_height\_vps\_in\_luma\_samples} \quad \text{(H-121)}$$

where pic_width_vps_in_luma_samples and pic_height_vps_in_luma_samples are found in the vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format( ) syntax structure in the VPS.

Each layer with nuh_layer_id equal to currLayerId conforming to a profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer and the CPB is understood to be the BPB:

a) The value of layerSizeInSamplesY shall be less than or equal to MaxLumaPs, where MaxLumaPs is specified in Table A.8 for the tier and level of the layer.

b) The value of pic_width_vps_in_luma_samples of the vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format( ) syntax structure in the VPS shall be less than or equal to Sqrt(MaxLumaPs * 8).

c) The value of pic_height_vps_in_luma_samples of the vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format( ) syntax structure in the VPS shall be less than or equal to Sqrt(MaxLumaPs * 8).

d) The value of max_vps_dec_pic_buffering_minus1[olsIdx][LayerIdxInVps[currLayerId]][HighestTid] shall be less than or equal to MaxDpbSize as derived by Equation A-2, with PicSizeInSamplesY being replaced with layerSizeInSamplesY, for the tier and level of the layer.

e) For level 5 and higher levels, the value of CtbSizeY for the layer shall be equal to 32 or 64.

f) The value of NumPicTotalCurr for each picture in the layer shall be less than or equal to 8.

g) When decoding each coded picture in the layer, the value of num_tile_columns_minus1 shall be less than MaxTileCols and num_tile_rows_minus1 shall be less than MaxTileRows, where MaxTileCols and MaxTileRows are specified in Table A.8 for the tier and level of the layer.

h) For the VCL HRD parameters of the layer, CpbSize[i] shall be less than or equal to CpbVclFactor * MaxCPB for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0][HighestTid], inclusive, where CpbSize[i] is specified in clause F.13.1 and MaxCPB is specified in Table A.8 for the tier and level of the layer in units of CpbVclFactor bits.

i) For the NAL HRD parameters of the layer, CpbSize[i] shall be less than or equal to CpbNalFactor * MaxCPB for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0][HighestTid], inclusive, where CpbSize[i] is specified in clause F.13.1 and MaxCPB is specified in Table A.8 for the tier and level of the layer in units of CpbNalFactor bits.

Table A.8 specifies the limits for each level of each tier for levels other than level 8.5.

A tier and level to which a layer in an output operation point associated with an OLS in a bitstream conforms are indicated by the syntax elements general_tier_flag and general_level_idc if OpTid of the output layer set is equal to

vps_max_sub_layer_minus1, and by the syntax elements sub_layer_tier_flag[OpTid] and sub_layer_level_idc[OpTid] otherwise, as follows:

– If the specified level is not level 8.5, general_tier_flag or sub_layer_tier_flag[OpTid] equal to 0 indicates conformance to the Main tier, and general_tier_flag or sub_layer_tier_flag[OpTid] equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8, and general_tier_flag and sub_layer_tier_flag[OpTid] shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A.8 marked with "-"). Otherwise (the specified level is level 8.5), it is a requirement of bitstream conformance that general_tier_flag and sub_layer_tier_flag[OpTid] shall be equal to 1 and the value 0 for general_tier_flag and sub_layer_tier_flag[OpTid] is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of general_tier_flag and sub_layer_tier_flag[OpTid].
– general_level_idc and sub_layer_level_idc[OpTid] shall be set equal to a value of 30 times the level number specified in Table A.8.

H.11.2.2 Profile-specific tier and level limits for the Scalable Main, Scalable Main 10 and scalable format range extensions profiles

The following is specified for expressing the constraints in this clause:

– The variable HbrFactor is set equal to 1.
– The variable BrVclFactor is set equal to CpbVclFactor * HbrFactor.
– The variable BrNalFactor is set equal to CpbNalFactor * HbrFactor.
– The variable MinCr is set equal to MinCrBase * MinCrScaleFactor ÷ HbrFactor, where MinCrBase is specified in Table A.9.

Each layer conforming to the Scalable Main or Scalable Main 10 profiles or a scalable format range extensions profile at a specified tier and level shall obey the following constraints for each conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer and the CPB is understood to be the BPB:

a) The nominal removal time of access unit n (with n greater than 0) from the CPB, as specified in clause F.13.2.3, shall satisfy the constraint that AuNominalRemovalTime[n] − AuCpbRemovalTime[n − 1] is greater than or equal to Max(layerSizeInSamplesY ÷ MaxLumaSr, fR), where layerSizeInSamplesY is the value of layerSizeInSamplesY for access unit n − 1 and MaxLumaSr is the value specified in Table A.9 that applies to access unit n − 1 for the tier and level of the layer.

b) The difference between consecutive output times of pictures in different access units, as specified in clause F.13.3.3 shall satisfy the constraint that DpbOutputInterval[n] is greater than or equal to Max(layerSizeInSamplesY ÷ MaxLumaSr, fR), where layerSizeInSamplesY is the value of layerSizeInSamplesY of access unit n and MaxLumaSr is the value specified in Table A.9 for access unit n for the tier and level of the layer, provided that access unit n is an access unit that has a picture that is output and is not the last of such access units.

c) The removal time of access unit 0 shall satisfy the constraint that the number of coded slice segments in access unit 0 is less than or equal to Min(Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0]) + MaxSliceSegmentsPerPicture * layerSizeInSamplesY / MaxLumaPs), MaxSliceSegmentsPerPicture), for the value of layerSizeInSamplesY of access unit 0, where MaxSliceSegmentsPerPicture, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9 for the tier and level of the layer.

d) The difference between consecutive CPB removal times of access units n and n − 1 (with n greater than 0) shall satisfy the constraint that the number of slice segments in access unit n is less than or equal to Min((Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1])), MaxSliceSegmentsPerPicture), where MaxSliceSegmentsPerPicture, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9 that apply to access unit n for the tier and level of the layer.

e) For the VCL HRD parameters for the layer, BitRate[i] shall be less than or equal to BrVclFactor * MaxBR for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0][HighestTid], inclusive, where BitRate[i] is specified in clause F.13.1 and MaxBR is specified in Table A.9 in units of BrVclFactor bits/s for the tier and level of the layer.

f) For the NAL HRD parameters for the layer, BitRate[i] shall be less than or equal to BrNalFactor * MaxBR for at least one of the delivery schedules identified by bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]] for combIdx ranging from 0 to num_bsp_schedules_minus1[olsIdx][0]

[HighestTid], inclusive, where BitRate[i] is specified in clause F.13.1 and MaxBR is specified in Table A.9 in units of BrNalFactor bits/s for the tier and level of the layer.

g) The sum of the NumBytesInNalUnit variables for access unit 0 shall be less than or equal to FormatCapabilityFactor * (Max(layerSizeInSamplesY, fR * MaxLumaSr) + MaxLumaSr * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0])) ÷ MinCr for the value of layerSizeInSamplesY of access unit 0, where MaxLumaSr is specified in Table A.9, and both MaxLumaSr and FormatCapabilityFactor are the values that apply to access unit 0 for the tier and level of the layer.

h) The sum of the NumBytesInNalUnit variables for access unit n (with n greater than 0) shall be less than or equal to FormatCapabilityFactor * MaxLumaSr * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1]) ÷ MinCr, where MaxLumaSr is specified in Table A.9, and both MaxLumaSr and FormatCapabilityFactor are the values that apply to access unit n for the tier and level of the layer.

i) The removal time of access unit 0 shall satisfy the constraint that the number of tiles in coded pictures in access unit 0 is less than or equal to Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[0] − AuNominalRemovalTime[0]) + MaxTileCols * MaxTileRows * PicSizeInSamplesY / MaxLumaPs), MaxTileCols * MaxTileRows), for the value of layerSizeInSamplesY of access unit 0, where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to access unit 0 for the tier and level of the layer.

j) The difference between consecutive CPB removal times of access units n and n − 1 (with n greater than 0) shall satisfy the constraint that the number of tiles in coded pictures in access unit n is less than or equal to Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[n] − AuCpbRemovalTime[n − 1])), MaxTileCols * MaxTileRows), where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to access unit n for the tier and level of the layer.

H.11.3 Decoder capabilities

When a decoder conforms to any profile specified in Annex H, it shall also have the INBLD capability specified in clause F.11.1.

Clause F.11.2 specifies requirements for a decoder conforming to any profile specified in this annex.

H.12 Byte stream format

The specifications in clause F.12 apply.

H.13 Hypothetical reference decoder

The specifications in clause F.13 and its subclauses apply.

H.14 Supplemental enhancement information

The specifications in Annex D and clause F.14 and their subclauses apply.

H.15 Video usability information

The specifications in clause F.15 and its subclauses apply.

Annex I

3D high efficiency video coding

(This annex forms an integral part of this Recommendation | International Standard.)

I.1 Scope

This annex specifies syntax, semantics, and decoding processes for 3D high efficiency video coding that use the syntax, semantics, and decoding processes specified in clauses 2-10 and Annexes A-G. This annex also specifies profiles, tiers, and levels for 3D high efficiency video coding.

I.2 Normative references

The list of normative references in clause G.2 apply.

I.3 Definitions

For the purpose of this annex, the following definitions apply in addition to the definitions in clause G.3. These definitions are either not present in clause G.3 or replace definitions in clause G.3.

I.3.1 **depth intra contour prediction**: A *prediction* of a *partition pattern* for a *prediction block* in a *picture* of a *depth layer* derived from samples of a *picture* included in the same *access unit* and in the *texture layer* of the same *view*.

I.3.2 **depth layer**: A *layer* with a nuh_layer_id value equal to i, such that DepthLayerFlag[i] is equal to 1 and DependencyId[i] and AuxId[i] are equal to 0.

I.3.3 **depth look-up table**: A list containing *depth values*.

I.3.4 **depth value**: A sample value of a *decoded picture* of a *depth layer*.

I.3.5 **disparity vector**: A *motion vector* used for inter-view prediction.

I.3.6 **inter-component prediction**: An *inter-layer prediction* where the *reference pictures* are associated with a DepthFlag value different from the DepthFlag value of the current *picture*.

I.3.7 **inter-view prediction**: An *inter-layer prediction* where the *reference pictures* are associated with *reference view order index* values different from the ViewIdx value of the current *picture*.

I.3.8 **intra prediction**: A *prediction* derived from only data elements (e.g., sample values) of the same decoded *slice* and additionally *may* be using *depth intra contour prediction*.

I.3.9 **partition pattern**: An MxM (M-column by M-row) array of *flags* defining two *sub-block partitions* of an MxM *prediction block*.

I.3.10 **prediction block**: A rectangular MxN *block* of samples on which either the same *prediction* or *partitioning* in *sub-block partitions* is applied.

I.3.11 **reference view order index**: A ViewIdx value associated with a *reference picture* used for *inter-view prediction*.

I.3.12 **sub-block partition**: A subset of samples of a *prediction block* on which the same *prediction* is applied.

I.3.13 **texture layer**: A *layer* with a nuh_layer_id value equal to i, such that DepthLayerFlag[i], DependencyId[i], and AuxId[i] are equal to 0.

I.4 Abbreviations

The specifications in clause G.4 apply.

I.5 Conventions

The specifications in clause G.5 apply.

I.6 Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships

I.6.1 Bitstream formats

The specifications in clause 6.1 apply.

I.6.2 Source, decoded, and output picture formats

The specifications in clause 6.2 apply.

I.6.3 Partitioning of pictures, slices, slice segments, tiles, CTUs, and CTBs

The specifications in clause 6.3 and its subclauses apply.

I.6.4 Availability processes

The specifications in clause 6.4 apply.

I.6.5 Scanning processes

The specifications in clause 6.5 and its subclauses apply.

I.6.6 Derivation process for a wedgelet partition pattern table

NOTE – Tables and values resulting from this process are independent of any information contained in the bitstream.

The list WedgePatternTable[log2BlkSize] of partition patterns of size (1 << log2BlkSize)x(1 << log2BlkSize) and the variable NumWedgePattern[log2BlkSize] specifying the number of partition patterns in list WedgePatternTable[log2BlkSize] are derived as follows:

- For log2BlkSize in the range of 2 to 4, inclusive, the following applies:
  - NumWedgePattern[log2BlkSize] is set equal to 0.
  - The variable resShift is set equal to (log2BlkSize = = 4) ? 0 : 1.
  - The variable wBlkSize is set equal to (1 << (log2BlkSize + resShift)).
  - For wedgeOri in the range of 0 to 5, inclusive, the following applies:
    - The variable posEnd is set equal to NumWedgePattern[log2BlkSize].
    - If wedgeOri is equal to 0 or 4, the following applies:
      - The variables sizeScaleS and sizeScaleE are derived as follows:

        sizeScaleS = (log2BlkSize > 3) ? 2 : 1 (I-1)

        sizeScaleE = (wedgeOri < 4 && log2BlkSize > 3) ? : 2 : 1 (I-2)

      - For m in the range of 0 to (wBlkSize / sizeScaleS − 1), inclusive, the following applies:
        - For n in the range of 0 to (wBlkSize / sizeScaleE − 1), inclusive, the following applies:
          - The wedgelet partition pattern generation process as specified in clause I.6.6.1 is invoked with patternSize equal to (1 << log2BlkSize), the variable resShift, the variable wedgeOri, the variable (xS, yS) equal to (m * sizeScaleS, 0), the variable (xE, yE) equal to (wedgeOri = = 0) ? (0, n * sizeScaleE) : (n * sizeScaleE, wBlkSize − 1) as inputs, and the output is the partition pattern curWedgePattern.
          - The wedgelet partition pattern table insertion process as specified in clause I.6.6.2 is invoked with the variable log2BlkSize and the partition pattern curWedgePattern as inputs.
    - Otherwise (wedgeOri is equal to 1, 2, 3, or 5), the following applies:
      - For curPos in the range of posStart to posEnd − 1, inclusive, the following applies:
        - The partition pattern curWedgePattern[x][y] is derived as follows:

```
for( y = 0; y < ( 1 << log2BlkSize ); y++ )
    for( x = 0; x < ( 1 << log2BlkSize ); x++ )                          (I-3)
        curWedgePattern[ x ][ y ] = 1 −
            WedgePatternTable[ log2BlkSize ][ curPos ][ y ][ ( 1 << log2BlkSize ) − 1 − x ]
```

    - The variable posStart is set equal to posEnd.
- NumWedgePattern[5] is set equal to NumWedgePattern[4].
- For k = 0..NumWedgePattern[5] − 1, the following applies:
  - For x, y = 0..(1 << 5) − 1, the following applies:

    WedgePatternTable[5][k][x][y] = WedgePatternTable[4][k][x >> 1][y >> 1] (I-4)

I.6.6.1 Wedgelet partition pattern generation process

Inputs to this process are:

- a variable patternSize specifying the partition pattern size,
- a variable resShift specifying the precision of the partition pattern start and end locations relative to patternSize,
- a variable wedgeOri specifying the orientation of the partition pattern,
- a location (xS, yS) specifying the boundary start of a sub-block partition,
- a location (xE, yE) specifying the boundary end of a sub-block partition.

Output of this process is the partition pattern wedgePattern[x][y] of size (patternSize)x(patternSize).

The values of the partition pattern wedgePattern[x][y] are derived as specified by the following ordered steps:

1. For x, y = 0..patternSize − 1, wedgePattern[x][y] is set equal to 0.
2. The samples of the partition pattern wedgePattern that form a line between (xS, yS) and (xE, yE) are set equal to 1 as follows:

```
( x0, y0 ) = ( xS, yS )
( x1, y1 ) = ( xE, yE )
if( abs( yE − yS ) > abs( xE − xS ) ) {
    ( x0, y0 ) = Swap( x0, y0 )
    ( x1, y1 ) = Swap( x1, y1 )
}
if( x0 > x1 ) {
    ( x0, x1 ) = Swap( x0, x1 )
    ( y0, y1 ) = Swap( y0, y1 )
}
sumErr = 0                                                              (I-5)
posY = y0
for( posX = x0; posX <= x1; posX++ ) {
    if( abs( yE − yS ) > abs( xE − xS ) )
        wedgePattern[ posY >> resShift ][ posX >> resShift ] = 1
    else
        wedgePattern[ posX >> resShift ][ posY >> resShift ] = 1
    sumErr += ( abs( y1 − y0 ) << 1 )
    if( sumErr >= ( x1 − x0 ) ) {
        posY += ( y0 < y1 ) ? 1 : −1
        sumErr −= ( x1 − x0 ) << 1
    }
}
```

3. The samples of wedgePattern are modified as follows:

```
for( y = 0; y <= ( yE >> resShift ); y++ )
    for( x = 0; ( x <= patternSize − 1 ) && ( wedgePattern[ x ][ y ] == 0 ); x++ )   (I-6)
        wedgePattern[ x ][ y ] = 1
```

I.6.6.2 Wedgelet partition pattern table insertion process

Inputs to this process are:

- a variable log2BlkSize specifying the partition pattern size,
- a partition pattern wedgePattern[x][y], with x, y = 0..(1 << log2BlkSize) − 1.

The variable validPatternFlag is set equal to 0 and the following applies:

1. For x, y = 0..(1 << log2BlkSize) − 1, the following applies:
   - When wedgePattern[x][y] is not equal to wedgePattern[0][0], validPatternFlag is set equal to 1.
2. For k = 0..NumWedgePattern[log2BlkSize] − 1, the following applies:
   - The variable patIdenticalFlag is set equal to 1.
   - For x, y = 0..(1 << log2BlkSize) − 1, the following applies:

  – When wedgePattern[x][y] is not equal to WedgePatternTable[log2BlkSize][k][x][y], patIdenticalFlag is set equal to 0.

 – When patIdenticalFlag is equal to 1, validPatternFlag is set equal to 0.

3. For k = 0..NumWedgePattern[log2BlkSize] − 1, the following applies:

 – The variable patInvIdenticalFlag is set equal to 1.

 – For x, y = 0..(1 << log2BlkSize) − 1, the following applies:

  – When wedgePattern[x][y] is equal to WedgePatternTable[log2BlkSize][k][x][y], patInvIdenticalFlag is set equal to 0.

 – When patInvIdenticalFlag is equal to 1, validPatternFlag is set equal to 0.

When validPatternFlag is equal to 1, the following applies:

– The pattern WedgePatternTable[log2BlkSize][NumWedgePattern[log2BlkSize]] is set equal to wedgePattern.

– The value of NumWedgePattern[log2BlkSize] is incremented by one.

I.7 Syntax and semantics

I.7.1 Method of specifying syntax in tabular form

The specifications in clause F.7.1 apply.

I.7.2 Specification of syntax functions, categories, and descriptors

The specifications in clause F.7.2 apply.

I.7.3 Syntax in tabular form

I.7.3.1 NAL unit syntax

The specifications in clause F.7.3.1 and all its subclauses apply.

I.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

I.7.3.2.1 Video parameter set RBSP

video_parameter_set_rbsp( ) {	**Descriptor**
vps_video_parameter_set_id	u(4)
vps_base_layer_internal_flag	u(1)
vps_base_layer_available_flag	u(1)
vps_max_layers_minus1	u(6)
vps_max_sub_layers_minus1	u(3)
vps_temporal_id_nesting_flag	u(1)
vps_reserved_0xffff_16bits	u(16)
profile_tier_level(1, vps_max_sub_layers_minus1)	
vps_sub_layer_ordering_info_present_flag	u(1)
for(i = (vps_sub_layer_ordering_info_present_flag ? 0 : vps_max_sub_layers_minus1); i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering_minus1[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase_plus1[i]	ue(v)
}	
vps_max_layer_id	u(6)
vps_num_layer_sets_minus1	ue(v)
for(i = 1; i <= vps_num_layer_sets_minus1; i++)	
for(j = 0; j <= vps_max_layer_id; j++)	
layer_id_included_flag[i][j]	u(1)

vps_timing_info_present_flag	u(1)
if(vps_timing_info_present_flag) {	
vps_num_units_in_tick	u(32)
vps_time_scale	u(32)
vps_poc_proportional_to_timing_flag	u(1)
if(vps_poc_proportional_to_timing_flag)	
vps_num_ticks_poc_diff_one_minus1	ue(v)
vps_num_hrd_parameters	ue(v)
for(i = 0; i < vps_num_hrd_parameters; i++) {	
hrd_layer_set_idx[i]	ue(v)
if(i > 0)	
cprms_present_flag[i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
}	
vps_extension_flag	u(1)
if(vps_extension_flag) {	
while(!byte_aligned( ))	
vps_extension_alignment_bit_equal_to_one	u(1)
vps_extension( )	
vps_extension2_flag	u(1)
if(vps_extension2_flag) {	
vps_3d_extension_flag	u(1)
if(vps_3d_extension_flag) {	
while(!byte_aligned( ))	
vps_3d_extension_alignment_bit_equal_to_one	u(1)
vps_3d_extension( )	
}	
vps_extension3_flag	u(1)
if(vps_extension3_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
}	
}	
rbsp_trailing_bits( )	
}	

I.7.3.2.1.1 Video parameter set extension syntax

The specifications in clause F.7.3.2.1.1 apply.

I.7.3.2.1.2 Representation format syntax

The specifications in clause F.7.3.2.1.2 apply.

I.7.3.2.1.3 DPB size syntax

The specifications in clause F.7.3.2.1.3 apply.

I.7.3.2.1.4 VPS VUI syntax

The specifications in clause F.7.3.2.1.4 apply.

I.7.3.2.1.5 Video signal info syntax

The specifications in clause F.7.3.2.1.5 apply.

I.7.3.2.1.6 VPS VUI bitstream partition HRD parameters syntax

The specifications in clause F.7.3.2.1.6 apply.

I.7.3.2.1.7 Video parameter set 3D extension syntax

vps_3d_extension( ) {	**Descriptor**
cp_precision	ue(v)
for(n = 1; n < NumViews; n++) {	
i = ViewOIdxList[n]	
num_cp[i]	u(6)
if(num_cp[i] > 0) {	
cp_in_slice_segment_header_flag[i]	u(1)
for(m = 0; m < num_cp[i]; m++) {	
cp_ref_voi[i][m]	ue(v)
if(!cp_in_slice_segment_header_flag[i]) {	
j = cp_ref_voi[i][m]	
vps_cp_scale[i][j]	se(v)
vps_cp_off[i][j]	se(v)
vps_cp_inv_scale_plus_scale[i][j]	se(v)
vps_cp_inv_off_plus_off[i][j]	se(v)
}	
}	
}	
}	
}	

I.7.3.2.2 Sequence parameter set RBSP syntax

I.7.3.2.2.1 General sequence parameter set RBSP syntax

The specifications in clause F.7.3.2.2.1 apply.

I.7.3.2.2.2 Sequence parameter set range extension syntax

The specifications in clause F.7.3.2.2.2 apply.

I.7.3.2.2.3 Sequence parameter set screen content coding extension syntax

The specifications in clause F.7.3.2.2.3 apply.

I.7.3.2.2.4 Sequence parameter set multilayer extension syntax

The specifications in clause F.7.3.2.2.4 apply.

I.7.3.2.2.5 Sequence parameter set 3D extension syntax

sps_3d_extension( ) {	**Descriptor**
for(d = 0; d <= 1; d++) {	
iv_di_mc_enabled_flag[d]	u(1)
iv_mv_scal_enabled_flag[d]	u(1)
if(d = = 0) {	
log2_ivmc_sub_pb_size_minus3[d]	ue(v)

iv_res_pred_enabled_flag[d]	u(1)
depth_ref_enabled_flag[d]	u(1)
vsp_mc_enabled_flag[d]	u(1)
dbbp_enabled_flag[d]	u(1)
} else {	
tex_mc_enabled_flag[d]	u(1)
log2_texmc_sub_pb_size_minus3[d]	ue(v)
intra_contour_enabled_flag[d]	u(1)
intra_dc_only_wedge_enabled_flag[d]	u(1)
cqt_cu_part_pred_enabled_flag[d]	u(1)
inter_dc_only_enabled_flag[d]	u(1)
skip_intra_enabled_flag[d]	u(1)
}	
}	
}	

I.7.3.2.3 Picture parameter set RBSP syntax

I.7.3.2.3.1 General picture parameter set RBSP syntax

The specifications in clause F.7.3.2.3.1 apply.

I.7.3.2.3.2 Picture parameter set range extension syntax

The specifications in clause F.7.3.2.3.2 apply.

I.7.3.2.3.3 Picture parameter set screen content coding extension syntax

The specifications in clause F.7.3.2.3.3 apply.

I.7.3.2.3.4 Picture parameter set multilayer extension syntax

The specifications in clause F.7.3.2.3.4 apply.

I.7.3.2.3.5 General colour mapping table syntax

The specifications in clause F.7.3.2.3.5 apply.

I.7.3.2.3.6 Colour mapping octants syntax

The specifications in clause F.7.3.2.3.6 apply.

I.7.3.2.3.7 Picture parameter set 3D extension syntax

pps_3d_extension( ) {	**Descriptor**
dlts_present_flag	u(1)
if(dlts_present_flag) {	
pps_depth_layers_minus1	u(6)
pps_bit_depth_for_depth_layers_minus8	u(4)
for(i = 0; i <= pps_depth_layers_minus1; i++) {	
dlt_flag[i]	u(1)
if(dlt_flag[i]) {	
dlt_pred_flag[i]	u(1)
if(!dlt_pred_flag[i])	
dlt_val_flags_present_flag[i]	u(1)
if(dlt_val_flags_present_flag[i])	
for(j = 0; j <= depthMaxValue; j++)	

dlt_value_flag[i][j]	u(1)
else	
delta_dlt(i)	
}	
}	
}	
}	

I.7.3.2.3.8 Delta depth look-up table syntax

delta_dlt(i) {	**Descriptor**
num_val_delta_dlt	u(v)
if(num_val_delta_dlt > 0) {	
if(num_val_delta_dlt > 1)	
max_diff	u(v)
if(num_val_delta_dlt > 2 && max_diff > 0)	
min_diff_minus1	u(v)
delta_dlt_val0	u(v)
if(max_diff > (min_diff_minus1 + 1))	
for(k = 1; k < num_val_delta_dlt; k++)	
delta_val_diff_minus_min[k]	u(v)
}	
}	

I.7.3.2.4 Supplemental enhancement information RBSP syntax

The specifications in clause F.7.3.2.4 apply.

I.7.3.2.5 Access unit delimiter RBSP syntax

The specifications in clause F.7.3.2.5 apply.

I.7.3.2.6 End of sequence RBSP syntax

The specifications in clause F.7.3.2.6 apply.

I.7.3.2.7 End of bitstream RBSP syntax

The specifications in clause F.7.3.2.7 apply.

I.7.3.2.8 Filler data RBSP syntax

The specifications in clause F.7.3.2.8 apply.

I.7.3.2.9 Slice segment layer RBSP syntax

The specifications in clause F.7.3.2.9 apply.

I.7.3.2.10 RBSP slice segment trailing bits syntax

The specifications in clause F.7.3.2.10 apply.

I.7.3.2.11 RBSP trailing bits syntax

The specifications in clause F.7.3.2.11 apply.

I.7.3.2.12 Byte alignment syntax

The specifications in clause F.7.3.2.12 apply.

I.7.3.3 Profile, tier and level syntax

The specifications in clause F.7.3.3 apply.

I.7.3.4 Scaling list data syntax

The specifications in clause F.7.3.4 apply.

I.7.3.5 Supplemental enhancement information message syntax

The specifications in clause F.7.3.5 apply.

I.7.3.6 Slice segment header syntax

I.7.3.6.1 General slice segment header syntax

slice_segment_header( ) {	**Descriptor**
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
if(!dependent_slice_segment_flag) {	
i = 0	
if(num_extra_slice_header_bits > i) {	
i++	
discardable_flag	u(1)
}	
if(num_extra_slice_header_bits > i) {	
i++	
cross_layer_bla_flag	u(1)
}	
for(; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag = = 1)	
colour_plane_id	u(2)
if((nuh_layer_id > 0 && !poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]]) || (nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP))	
slice_pic_order_cnt_lsb	u(v)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
st_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	

if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && !default_ref_layers_active_flag && NumRefListLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumRefListLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumRefListLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(inCmpPredAvailFlag)	
in_comp_pred_flag	u(1)
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(slice_type = = P || slice_type = = B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type = = B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification( )	
if(slice_type = = B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	

cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type = = P) || (weighted_bipred_flag && slice_type = = B))	
pred_weight_table( )	
else if(!DepthFlag && NumRefListLayers[nuh_layer_id] > 0) {	
slice_ic_enabled_flag	u(1)
if(slice_ic_enabled_flag)	
slice_ic_disabled_merge_zero_idx_flag	u(1)
}	
five_minus_max_num_merge_cand	ue(v)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag || slice_sao_chroma_flag || !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if(cp_in_slice_segment_header_flag[ViewIdx])	
for(m = 0; m < num_cp[ViewIdx]; m++) {	
j = cp_ref_voi[ViewIdx][m]	
cp_scale[j]	se(v)
cp_off[j]	se(v)
cp_inv_scale_plus_scale[j]	se(v)
cp_inv_off_plus_off[j]	se(v)
}	
}	
if(tiles_enabled_flag || entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)

if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
if(poc_reset_info_present_flag)	
poc_reset_idc	u(2)
if(poc_reset_idc != 0)	
poc_reset_period_id	u(6)
if(poc_reset_idc = = 3) {	
full_poc_reset_flag	u(1)
poc_lsb_val	u(v)
}	
if(!PocMsbValRequiredFlag && vps_poc_lsb_aligned_flag)	
poc_msb_cycle_val_present_flag	u(1)
if(poc_msb_cycle_val_present_flag)	
poc_msb_cycle_val	ue(v)
while(more_data_in_slice_segment_header_extension( ))	
slice_segment_header_extension_data_bit	u(1)
}	
byte_alignment( )	
}	

I.7.3.6.2 Reference picture list modification syntax

The specifications in clause F.7.3.6.2 apply.

I.7.3.6.3 Weighted prediction parameters syntax

The specifications in clause F.7.3.6.3 apply.

I.7.3.7 Short-term reference picture set syntax

The specifications in clause F.7.3.7 apply.

I.7.3.8 Slice segment data syntax

I.7.3.8.1 General slice segment data syntax

The specifications in clause F.7.3.8.1 apply.

I.7.3.8.2 Coding tree unit syntax

The specifications in clause F.7.3.8.2 apply.

I.7.3.8.3 Sample adaptive offset syntax

The specifications in clause F.7.3.8.3 apply.

I.7.3.8.4 Coding quadtree syntax

coding_quadtree(x0, y0, log2CbSize, cqtDepth) {	**Descriptor**
if(x0 + (1 << log2CbSize) <= pic_width_in_luma_samples && y0 + (1 << log2CbSize) <= pic_height_in_luma_samples && log2CbSize > MinCbLog2SizeY && !predSplitCuFlag)	
split_cu_flag[x0][y0]	ae(v)
if(cu_qp_delta_enabled_flag && log2CbSize >= Log2MinCuQpDeltaSize) {	
IsCuQpDeltaCoded = 0	
CuQpDeltaVal = 0	
}	
if(cu_chroma_qp_offset_enabled_flag &&	
log2CbSize >= Log2MinCuChromaQpOffsetSize)	
IsCuChromaQpOffsetCoded = 0	
if(split_cu_flag[x0][y0]) {	
x1 = x0 + (1 << (log2CbSize − 1))	
y1 = y0 + (1 << (log2CbSize − 1))	
coding_quadtree(x0, y0, log2CbSize − 1, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples)	
coding_quadtree(x1, y0, log2CbSize − 1, cqtDepth + 1)	
if(y1 < pic_height_in_luma_samples)	
coding_quadtree(x0, y1, log2CbSize − 1, cqtDepth + 1)	
if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)	
coding_quadtree(x1, y1, log2CbSize − 1, cqtDepth + 1)	
} else	
coding_unit(x0, y0, log2CbSize)	
}	

I.7.3.8.5 Coding unit syntax

coding_unit(x0, y0, log2CbSize) {	**Descriptor**
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)
nCbS = (1 << log2CbSize)	
if(cu_skip_flag[x0][y0])	
prediction_unit(x0, y0, nCbS, nCbS)	
else if(SkipIntraEnabledFlag)	
skip_intra_flag[x0][y0]	ae(v)
if(!cu_skip_flag[x0][y0] && !skip_intra_flag[x0][y0]) {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if((CuPredMode[x0][y0] != MODE_INTRA || log2CbSize = = MinCbLog2SizeY) && !predPartModeFlag)	
part_mode	ae(v)
if(CuPredMode[x0][y0] = = MODE_INTRA) {	

if(PartMode = = PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	f(1)
pcm_sample(x0, y0, log2CbSize)	
} else {	
pbOffset = (PartMode = = PART_NxN) ? (nCbS / 2) : nCbS	
log2PbSize = log2CbSize − ((PartMode = = PART_NxN) ? 1 : 0)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset) {	
if(IntraDcOnlyWedgeEnabledFlag || IntraContourEnabledFlag)	
intra_mode_ext(x0 + i , y0 + j , log2PbSize)	
if(no_dim_flag[x0 + i][y0 + j])	
prev_intra_luma_pred_flag[x0 + i][y0 + j]	ae(v)
}	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
if(no_dim_flag[x0 + i][y0 + j]) {	
if(prev_intra_luma_pred_flag[x0 + i][y0 + j])	
mpm_idx[x0 + i][y0 + j]	ae(v)
else	
rem_intra_luma_pred_mode[x0 + i][y0 + j]	ae(v)
}	
if(ChromaArrayType = = 3)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
intra_chroma_pred_mode[x0 + 1][y0 + j]	ae(v)
else if(ChromaArrayType != 0)	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	
} else {	
if(PartMode = = PART_2Nx2N)	
prediction_unit(x0, y0, nCbS, nCbS)	
else if(PartMode = = PART_2NxN) {	
prediction_unit(x0, y0, nCbS, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS, nCbS / 2)	
} else if(PartMode = = PART_Nx2N) {	
prediction_unit(x0, y0, nCbS / 2, nCbS)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS)	
} else if(PartMode = = PART_2NxnU) {	
prediction_unit(x0, y0, nCbS, nCbS / 4)	
prediction_unit(x0, y0 + (nCbS / 4), nCbS, nCbS * 3 / 4)	
} else if(PartMode = = PART_2NxnD) {	
prediction_unit(x0, y0, nCbS, nCbS * 3 / 4)	
prediction_unit(x0, y0 + (nCbS * 3 / 4), nCbS, nCbS / 4)	
} else if(PartMode = = PART_nLx2N) {	

prediction_unit(x0, y0, nCbS / 4, nCbS)	
prediction_unit(x0 + (nCbS / 4), y0, nCbS * 3 / 4, nCbS)	
} else if(PartMode = = PART_nRx2N) {	
prediction_unit(x0, y0, nCbS * 3 / 4, nCbS)	
prediction_unit(x0 + (nCbS * 3 / 4), y0, nCbS / 4, nCbS)	
} else { /* PART_NxN */	
prediction_unit(x0, y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
}	
}	
}	
cu_extension(x0, y0, log2CbSize)	
if(DcOnlyFlag[x0][y0] | | (!skip_intra_flag[x0][y0] && CuPredMode[x0][y0] = = MODE_INTRA))	
depth_dcs(x0, y0, log2CbSize)	
if(!cu_skip_flag[x0][y0] && !skip_intra_flag[x0][y0] && !dc_only_flag[x0][y0] && !pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA && !(PartMode = = PART_2Nx2N && merge_flag[x0][y0]))	
rqt_root_cbf	ae(v)
if(rqt_root_cbf) {	
MaxTrafoDepth = (CuPredMode[x0][y0] = = MODE_INTRA ? (max_transform_hierarchy_depth_intra + IntraSplitFlag) : max_transform_hierarchy_depth_inter)	
transform_tree(x0, y0, x0, y0, log2CbSize, 0, 0)	
}	
}	
}	

I.7.3.8.5.1 Intra mode extension syntax

intra_mode_ext(x0, y0, log2PbSize) {	**Descriptor**
if(log2PbSize < 6)	
no_dim_flag[x0][y0]	ae(v)
if(!no_dim_flag[x0][y0] && IntraDcOnlyWedgeEnabledFlag && IntraContourEnabledFlag)	
depth_intra_mode_idx_flag[x0][y0]	ae(v)
if(!no_dim_flag[x0][y0] && !depth_intra_mode_idx_flag[x0][y0])	
wedge_full_tab_idx[x0][y0]	ae(v)
}	

I.7.3.8.5.2 Coding unit extension syntax

cu_extension(x0, y0, log2CbSize) {	**Descriptor**
if(skip_intra_flag[x0][y0])	
skip_intra_mode_idx[x0][y0]	ae(v)
else {	
if(!cu_skip_flag[x0][y0]) {	
if(DbbpEnabledFlag && DispAvailFlag && log2CbSize > 3 && (PartMode = = PART_2NxN || PartMode = = PART_Nx2N))	
dbbp_flag[x0][y0]	ae(v)
if((CuPredMode[x0][y0] = = MODE_INTRA ? IntraDcOnlyWedgeEnabledFlag : InterDcOnlyEnabledFlag) && PartMode = = PART_2Nx2N)	
dc_only_flag[x0][y0]	ae(v)
}	
if(CuPredMode[x0][y0] != MODE_INTRA && PartMode = = PART_2Nx2N) {	
if(IvResPredEnabledFlag && RpRefPicAvailFlag)	
iv_res_pred_weight_idx[x0][y0]	ae(v)
if(slice_ic_enabled_flag && icCuEnableFlag && iv_res_pred_weight_idx[x0][y0] = = 0)	
illu_comp_flag[x0][y0]	ae(v)
}	
}	
}	

I.7.3.8.5.3 Depth DCs syntax

depth_dcs(x0, y0, log2CbSize) {	**Descriptor**
nCbS = (1 << log2CbSize)	
pbOffset = (PartMode = = PART_NxN && CuPredMode[x0][y0] = = MODE_INTRA) ? (nCbS / 2) : nCbS	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(k = 0; k < nCbS; k = k + pbOffset)	
if(DimFlag[x0 + k][y0 + j] || DcOnlyFlag[x0][y0]) {	
if(CuPredMode[x0][y0] = = MODE_INTRA && DcOnlyFlag[x0][y0])	
depth_dc_present_flag[x0 + k][y0 + j]	ae(v)
dcNumSeg = DimFlag[x0 + k][y0 + j] ? 2 : 1	
if(depth_dc_present_flag[x0 + k][y0 + j])	
for(i = 0; i < dcNumSeg; i++) {	
depth_dc_abs[x0 + k][y0 + j][i]	ae(v)
if((depth_dc_abs[x0 + k][y0 + j][i] − dcNumSeg + 2) > 0)	
depth_dc_sign_flag[x0 + k][y0 + j][i]	ae(v)
}	
}	
}	

I.7.3.8.6 Prediction unit syntax

The specifications in clause F.7.3.8.6 apply.

I.7.3.8.7 PCM sample syntax

The specifications in clause F.7.3.8.7 apply.

I.7.3.8.8 Transform tree syntax

The specifications in clause F.7.3.8.8 apply.

I.7.3.8.9 Motion vector difference coding syntax

The specifications in clause F.7.3.8.9 apply.

I.7.3.8.10 Transform unit syntax

The specifications in clause F.7.3.8.10 apply.

I.7.3.8.11 Residual coding syntax

The specifications in clause F.7.3.8.11 apply.

I.7.3.8.12 Cross-component prediction syntax

The specifications in clause F.7.3.8.12 apply.

I.7.3.8.13 Palette mode syntax

The specifications in clause F.7.3.8.13 apply

I.7.3.8.14 Delta QP syntax

The specifications in clause F.7.3.8.14 apply.

I.7.3.8.15 Chroma QP offset syntax

The specifications in clause F.7.3.8.15 apply.

I.7.4 Semantics

I.7.4.1 General

I.7.4.2 NAL unit semantics

I.7.4.2.1 General NAL unit semantics

The specifications in clause F.7.4.2.1 apply.

I.7.4.2.2 NAL unit header semantics

The specifications in clause F.7.4.2.2 apply.

I.7.4.2.3 Encapsulation of an SODB within an RBSP (informative)

The specifications in clause F.7.4.2.3 apply.

I.7.4.2.4 Order of NAL units and association to coded pictures, access units, and coded video sequences

The specifications in clause F.7.4.2.4 and all its subclauses apply.

I.7.4.3 Raw byte sequence payloads, trailing bits, and byte alignment semantics

I.7.4.3.1 Video parameter set RBSP semantics

The specifications in clause F.7.4.3.1 apply with the following modifications and additions:

vps_extension2_flag equal to 0 specifies that no vps_3d_extension( ) syntax structure and no vps_extension_data_flag syntax elements are present in the VPS RBSP syntax structure. vps_extension2_flag equal to 1 specifies that the vps_3d_extension( ) syntax structure and vps_extension_data_flag syntax elements may be present in the VPS RBSP syntax structure. When MaxLayersMinus1 is greater than 0, vps_extension2_flag shall be equal to 1.

vps_3d_extension_flag equal to 0 specifies that no vps_3d_extension( ) syntax structure is present in the VPS RBSP syntax structure. vps_3d_extension_flag equal to 1 specifies that the vps_3d_extension( ) syntax structure is present in the VPS RBSP syntax structure. When MaxLayersMinus1 is greater than 0, vps_3d_extension_flag shall be equal to 1.

vps_3d_extension_alignment_bit_equal_to_one shall be equal to 1.

vps_extension3_flag equal to 0 specifies that no vps_extension_data_flag syntax elements are present in the VPS RBSP syntax structure. vps_extension3_flag shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for vps_extension3_flag is reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this version of this Specification shall ignore all data that follow the value 1 for vps_extension3_flag in a VPS RBSP.

vps_extension_data_flag may have any value. Its presence and value do not affect decoder conformance to profiles specified in Annexes A, G, H, or I. Decoders conforming to a profile specified in Annexes A, G, H, or I shall ignore all vps_extension_data_flag syntax elements.

I.7.4.3.1.1 Video parameter set extension semantics

The specifications in clause F.7.4.3.1.1 apply with the following additions and modifications:

direct_dependency_type[i][j] indicates the type of dependency between the layer predLayer with nuh_layer_id equal layer_id_in_nuh[i] and the layer refLayer with nuh_layer_id equal to layer_id_in_nuh[j]. ((direct_dependency_type[i][j] +1) & 0x1) greater than 0 specifies that samples of refLayer may be used for inter-layer prediction of predLayer. ((direct_dependency_type[i][j] +1) & 0x1) equal to 0 specifies that samples of refLayer are not used for inter-layer prediction of predLayer. ((direct_dependency_type[i][j] +1) & 0x2) greater than 0 specifies that motion vectors of refLayer may be used for inter-layer prediction of predLayer. ((direct_dependency_type[i][j] +1) & 0x2) equal to 0 specifies that motion vectors of refLayer are not used for inter-layer prediction of predLayer. ((direct_dependency_type[i][j] +1) & 0x4) greater than 0 specifies that coding quadtree and coding unit partitioning information of refLayer may be used for inter-layer prediction of predLayer. ((direct_dependency_type[i][j] +1) & 0x4) equal to 0 specifies that coding quadtree and coding unit partitioning information of refLayer are not used for inter-layer prediction of the predLayer.

The length of the direct_dependency_type[i][j] syntax element is direct_dep_type_len_minus2 + 2 bits. Although the value of direct_dependency_type[i][j] shall be in the range of 0 to 2, inclusive, when predLayer conforms to a profile specified in Annexes A, G, or H, and in the range of 0 to 6, inclusive, when the predLayer conforms to a profile specified in Annex I, decoders shall allow values of direct_dependency_type[i][j] in the range of 0 to $2^{32} - 2$, inclusive, to appear in the syntax.

The list ViewOIdxList[idx] is derived as follows:

```
idx = 0
ViewOIdxList[ idx++ ] = 0
for( i = 1; i <= MaxLayersMinus1; i++ ) {
    newViewFlag = 1
    for( j = 0; j < i; j++ )                                                    (I-7)
        if( ViewOrderIdx[ layer_id_in_nuh[ i ] ] = = ViewOrderIdx[ layer_id_in_nuh[ j ] ] )
            newViewFlag = 0
    if( newViewFlag )
        ViewOIdxList[ idx++ ] = ViewOrderIdx[ lId ]
}
```

The variables NumRefListLayers[iNuhLId] and IdRefListLayer[iNuhLId] are derived as follows:

```
for( i = 0; i <= MaxLayersMinus1; i++ ) {
    iNuhLId = layer_id_in_nuh[ i ]
    NumRefListLayers[ iNuhLId ] = 0
    for( j = 0; j < NumDirectRefLayers[ iNuhLId ]; j++ ) {
        jNuhLId = IdDirectRefLayer[ iNuhLId ][ j ]                              (I-8)
        if( DepthLayerFlag[ iNuhLId ] = = DepthLayerFlag[ jNuhLId ] )
            IdRefListLayer[ iNuhLId ][ NumRefListLayers[ iNuhLId ]++ ] = jNuhLId
    }
}
```

The variables ViewCompLayerPresentFlag[iViewOIdx][depFlag] and ViewCompLayerId[iViewOIdx][depFlag] are derived as follows:

```
for( depFlag = 0; depFlag <= 1; depFlag++ )
    for( i = 0; i < NumViews; i++ ) {
        iViewOIdx = ViewOIdxList[ i ]
        layerId = −1
        for( j = 0; j <= MaxLayersMinus1; j++ ) {
            jNuhLId = layer_id_in_nuh[ j ]                                      (I-9)
            if( DepthLayerFlag[ jNuhLId ] = = depFlag && ViewOrderIdx[ jNuhLId ] = = iViewOIdx
                && DependencyId[ jNuhLId ] = = 0 && AuxId[ jNuhLId ] = = 0 )
                layerId = jNuhLId
```

```
        }
        ViewCompLayerPresentFlag[ iViewOIdx ][ depFlag ] = ( layerId != −1 )
        ViewCompLayerId[ iViewOIdx ][ depFlag ] = layerId
    }
```

The function ViewIdx(picX) is specified as follows:

ViewIdx(picX) = ViewIdx of the picture picX (I-10)

The function ViewIdVal(picX) is specified as follows:

ViewIdVal(picX) = view_id_val[ViewIdx(picX)] (I-11)

I.7.4.3.1.2 Representation format semantics

The specifications in clause F.7.4.3.1.2 apply.

I.7.4.3.1.3 DPB size semantics

The specifications in clause F.7.4.3.1.3 apply.

I.7.4.3.1.4 VPS VUI semantics

The specifications in clause F.7.4.3.1.4 apply.

I.7.4.3.1.5 Video signal info semantics

The specifications in clause F.7.4.3.1.5 apply.

I.7.4.3.1.6 VPS VUI bitstream partition HRD parameters semantics

The specifications in clause F.7.4.3.1.6 apply.

I.7.4.3.1.7 Video parameter set 3D extension semantics

cp_precision + $\text{BitDepth}_Y - 1$ specifies the precision of the vps_cp_scale[i][j] and vps_cp_inv_scale_plus_scale[i][j] syntax elements present in the VPS and the cp_scale[j] and cp_inv_scale_plus_scale[j] syntax elements present in slice headers. The value of cp_precision shall be in the range of 0 to 5, inclusive.

num_cp[i], when cp_in_slice_segment_header_flag[i] is equal to 0, specifies the number of vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] syntax elements present for the view with ViewIdx equal to i in the VPS. num_cp[i], when cp_in_slice_segment_header_flag[i] is equal to 1, specifies the number of cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] syntax elements present in slice headers of layers with ViewIdx equal to i.

cp_in_slice_segment_header_flag[i] equal to 1 specifies that the syntax elements vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] for the view with ViewIdx equal to i are not present in the VPS and that the syntax elements cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] may be present in slice headers of layers with ViewIdx equal to i. cp_in_slice_segment_header_flag[i] equal to 0 specifies that the vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] syntax elements for the view with ViewIdx equal to i are present in the VPS and that the syntax elements cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] are not present in slice headers of layers with ViewIdx equal to i. When not present, the value of cp_in_slice_segment_header_flag[i] is inferred to be equal to 0.

cp_ref_voi[i][m], when cp_in_slice_segment_header_flag[i] is equal to 0, specifies the ViewIdx value j of the view to which the m-th vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] syntax element present for the view with ViewIdx equal to i in the VPS is related to. cp_ref_voi[i][m], when cp_in_slice_segment_header_flag[i] is equal to 1, specifies the ViewIdx value j of the view to which the m-th cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] syntax element present in the slice headers of layers with ViewIdx equal to i is related to. The value of cp_ref_voi[i][m] shall be in the range of 0 to 65 535, inclusive. It is a requirement of bitstream conformance that cp_ref_voi[i][x] is not equal to cp_ref_voi[i][y] for any values of x and y in the range of 0 to num_cp[i] − 1, inclusive, when x is not equal to y.

For n and m in the range of 0 to NumViews − 1, inclusive, the variable CpPresentFlag[ViewOIdxList[n]][ViewOIdxList[m]] is set equal to 0 and modified as follows:

```
for( n = 1; n < NumViews; n++ ) {
    i = ViewOIdxList[ n ]
    for( m = 0; m < num_cp[ i ]; m++ )                              (I-12)
        CpPresentFlag[ i ][ cp_ref_voi[ i ][ m ] ] = 1
}
```

vps_cp_scale[i][j], **vps_cp_off**[i][j], **vps_cp_inv_scale_plus_scale**[i][j], and **vps_cp_inv_off_plus_off**[i][j] specify parameters for derivation of a horizontal component of a disparity vector from a depth value and may be used to infer the values of the cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] syntax elements in slice headers of layers with ViewIdx equal to i. When both a texture layer and a depth layer with ViewIdx equal to i are present, the conversion parameters are associated with the texture layer with ViewIdx equal to i.

I.7.4.3.2 Sequence parameter set RBSP semantics

I.7.4.3.2.1 General sequence parameter set RBSP semantics

The specifications in clause F.7.4.3.2.1 apply.

I.7.4.3.2.2 Sequence parameter set range extension semantics

The specifications in clause F.7.4.3.2.2 apply.

I.7.4.3.2.1 Sequence parameter set screen content coding extension semantics

The specifications in clause F.7.4.3.2.3 apply.

I.7.4.3.2.2 Sequence parameter set multilayer extension semantic

The specifications in clause F.7.4.3.2.4 apply.

I.7.4.3.2.3 Sequence parameter set 3D extension semantics

iv_di_mc_enabled_flag[d] equal to 1 specifies that the derivation process for inter-view predicted merging candidates and the derivation process for disparity information merging candidates may be used in the decoding process of layers with DepthFlag equal to d. iv_di_mc_enabled_flag[d] equal to 0 specifies that derivation process for inter-view predicted merging candidates and the derivation process for disparity information merging candidates is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of iv_di_mc_enabled_flag[d] is inferred to be equal to 0.

iv_mv_scal_enabled_flag[d] equal to 1 specifies that motion vectors used for inter-view prediction may be scaled based on view_id_val values in the decoding process of layers with DepthFlag equal to d. iv_mv_scal_enabled_flag[d] equal to 0 specifies that motion vectors used for inter-view prediction are not scaled based on view_id_val values in the decoding process of layers with DepthFlag equal to d. When not present, the value of iv_mv_scal_enabled_flag[d] is inferred to be equal to 0.

log2_ivmc_sub_pb_size_minus3[d], when iv_di_mc_enabled_flag[d] is equal to 1 and d is equal to 0, is used to derive the minimum size of sub-block partitions used in the derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate in the decoding process of layers with DepthFlag equal to d. When not present, the value of log2_ivmc_sub_pb_size_minus3[d] is inferred to be equal to (CtbLog2SizeY − 3). The value of log2_ivmc_sub_pb_size_minus3[d] shall be in the range of (MinCbLog2SizeY − 3) to (CtbLog2SizeY − 3), inclusive.

iv_res_pred_enabled_flag[d] equal to 1 specifies that the iv_res_pred_weight_idx syntax element may be present in coding units of layers with DepthFlag equal to d. iv_res_pred_enabled_flag[d] equal to 0 specifies that the iv_res_pred_weight_idx syntax element is not present coding units of layers with DepthFlag equal to d. When not present, the value of iv_res_pred_enabled_flag[d] is inferred to be equal to 0.

vsp_mc_enabled_flag[d] equal to 1 specifies that the derivation process for a view synthesis prediction merging candidate may be used in the decoding process of layers with DepthFlag equal to d. vsp_mc_enabled_flag[d] equal to 0 specifies that the derivation process for a view synthesis prediction merging candidate is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of vsp_mc_enabled_flag[d] is inferred to be equal to 0.

dbbp_enabled_flag[d] equal to 1 specifies that the dbbp_flag syntax element may be present in coding units of layers with DepthFlag equal to d. dbbp_enabled_flag[d] equal to 0 specifies that the dbbp_flag syntax element is not present coding units of layers with DepthFlag equal to d. When not present, the value of dbbp_enabled_flag[d] is inferred to be equal to 0.

depth_ref_enabled_flag[d] equal to 1 specifies that the derivation process for a depth or disparity sample array from a depth picture may be used in the derivation process for a disparity vector for texture layers in the decoding process of layers with DepthFlag equal to d. depth_ref_enabled_flag[d] equal to 0 specifies that derivation process for a depth or disparity sample array from a depth picture is not used in the derivation process for a disparity vector for texture layers in the decoding process of layers with DepthFlag equal to d. When not present, the value of depth_ref_enabled_flag[d] is inferred to be equal to 0.

tex_mc_enabled_flag[d] equal to 1 specifies that the derivation process for motion vectors for the texture merge candidate may be used in the decoding process of layers with DepthFlag equal to d. tex_mc_enabled_flag[d] equal to 0

specifies that the derivation process for motion vectors for the texture merge candidate is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of tex_mc_enabled_flag[d] is inferred to be equal to 0.

log2_texmc_sub_pb_size_minus3[d], when tex_mc_enabled_flag[d] is equal to 1, is used to derive the minimum size of sub-block partitions used in the derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate in the decoding process of layers with DepthFlag equal to d. The value of log2_texmc_sub_pb_size_minus3[layerId] shall be in the range of (MinCbLog2SizeY − 3) to (CtbLog2SizeY − 3), inclusive.

intra_contour_enabled_flag[d] equal to 1 specifies that the intra prediction mode INTRA_CONTOUR using depth intra contour prediction may be used in the decoding process of layers with DepthFlag equal to d. intra_contour_enabled_flag[d] equal to 0 specifies that the intra prediction mode INTRA_CONTOUR using depth intra contour prediction is not used in the decoding process of layers with DepthFlag equal to d. When not present, intra_contour_enabled_flag[d] is inferred to be equal to 0.

intra_dc_only_wedge_enabled_flag[d] equal to 1 specifies that the dc_only_flag syntax element may be present in coding units coded in an intra prediction mode of layers with DepthFlag equal to d, and that the intra prediction mode INTRA_WEDGE may be used in the decoding process of layers with DepthFlag equal to d. intra_dc_only_wedge_enabled_flag[d] equal to 0 specifies that the dc_only_flag syntax element is not present in coding units coded in an intra prediction mode of layers with DepthFlag equal to d and that the intra prediction mode INTRA_WEDGE is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of intra_dc_only_wedge_enabled_flag[d] is inferred to be equal to 0.

cqt_cu_part_pred_enabled_flag[d] equal to 1 specifies that coding quadtree and coding unit partitioning information may be inter-component predicted in the decoding process of layers with DepthFlag equal to d. cqt_cu_part_pred_enabled_flag[d] equal to 0 specifies that coding quadtree and coding unit partitioning information are not inter-component predicted in the decoding process of layers with DepthFlag equal to d. When not present, the value of cqt_cu_part_pred_enabled_flag[d] is inferred to be equal to 0.

inter_dc_only_enabled_flag[d] equal to 1 specifies that the dc_only_flag syntax element may be present in coding units coded an in inter prediction mode of layers with DepthFlag equal to d. inter_dc_only_enabled_flag[d] equal to 0 specifies that the dc_only_flag syntax element is not present in coding units coded in an inter prediction mode of layers with DepthFlag equal to d. When not present, the value of inter_dc_only_enabled_flag[layerId] is inferred to be equal to 0.

skip_intra_enabled_flag[d] equal to 1 specifies that the skip_intra_flag syntax element may be present in coding units of layers with DepthFlag equal to d. skip_intra_enabled_flag[d] equal to 0 specifies that the skip_intra_flag syntax element is not present in coding units of layers with DepthFlag equal to d. When not present, the value of skip_intra_enabled_flag[layerId] is inferred to be equal to 0.

I.7.4.3.3 Picture parameter set RBSP semantics

I.7.4.3.3.1 General picture parameter set RBSP semantics

The specifications in clause F.7.4.3.3.1 apply.

I.7.4.3.3.2 Picture parameter set range extension semantics

The specifications in clause F.7.4.3.3.2 apply.

I.7.4.3.3.3 Picture parameter set screen content coding extension semantics

The specifications in clause F.7.4.3.3.3 apply.

I.7.4.3.3.4 Picture parameter set multilayer extension semantics

The specifications in clause F.7.4.3.3.4 apply.

I.7.4.3.3.5 General colour mapping table semantics

The specifications in clause F.7.4.3.3.5 apply.

I.7.4.3.3.6 Colour mapping octants semantics

The specifications in clause F.7.4.3.3.6 apply.

I.7.4.3.3.7 Picture parameter set 3D extension semantics

dlts_present_flag equal to 1 specifies that syntax elements for the derivation of depth look-up tables are present in the PPS. dlts_present_flag equal to 0 specifies that syntax elements for the derivation of depth look-up tables are not present in the PPS.

The variables NumDepthLayers and DepIdxToLId[j] are derived as follows:

```
j = 0
for( i = 0; i  <=  MaxLayersMinus1; i++ ) {
    layerId = layer_id_in_nuh[ i ]
    if( DepthLayerFlag[ layerId ] )                                      (I-13)
        DepIdxToLId[ j++ ] = layerId
}
NumDepthLayers = j
```

pps_depth_layers_minus1 plus 1 specifies the number of depth layers. pps_depth_layers_minus1 shall be equal to NumDepthLayers − 1.

pps_bit_depth_for_depth_layers_minus8 plus 8 specifies the bit depth of the samples in depth layers. It is a requirement of bitstream conformance that pps_bit_depth_for_depth_layers_minus8 shall be equal to bit_depth_luma_minus8 of the SPS the current PPS refers to.

The variable depthMaxValue is set equal to (1 << (pps_bit_depth_for_depth_layers_minus8 + 8)) − 1.

dlt_flag[i] equal to 1 specifies that a depth look-up table for the layer with nuh_layer_id equal to DepIdxToLId[i] is present in the PPS and used for the decoding of the layer with nuh_layer_id equal to DepIdxToLId[i]. dlt_flag[i] equal to 0 specifies that a depth look-up table is not present for the layer with nuh_layer_id equal to DepIdxToLId[i]. When not present, the value of dlt_flag[i] is inferred to be equal to 0.

For i in the range of 0 to NumDepthLayers − 1, inclusive, the variable DltFlag[DepIdxToLId[i]] is set equal to dlt_flag[i].

dlt_pred_flag[i] equal to 1 indicates that the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLId[i] is predicted from the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLId[0]. dlt_pred_flag[i] equal to 0 indicates that the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLId[i] is not predicted from any other depth look-up table. The value of dlt_pred_flag[0] shall be equal to 0. It is a requirement of bitstream conformance that, when dlt_flag[0] is equal to 0, dlt_pred_flag[i] shall be equal to 0.

dlt_val_flags_present_flag[i] equal to 1 specifies the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLId[i] is derived from dlt_value_flag[i][j] syntax elements. dlt_val_flags_present_flag[i] equal to 0 specifies the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLId[i] is derived from the delta_dlt( ) syntax structure. When not present, the value of dlt_val_flags_present_flag[i] is inferred to be equal to 0.

dlt_value_flag[i][j] equal to 1 specifies that j is an entry in the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLId[i]. dlt_value_flag[i][j] equal to 0 specifies that j is not an entry in the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLId[i].

When dlt_val_flags_present_flag[i] is equal to 1, the following applies:

– The variable layerId is set equal to DepIdxToLId[i].

– The variables DltVal[layerId][n] and NumValDlt[layerId] of the depth look-up table of the layer with nuh_layer_id equal to layerId are derived as follows:

```
for( n = 0, j = 0; j  <=  depthMaxValue; j++ )
    if( dlt_value_flag[ i ][ j ] )                                       (I-14)
        DltVal[ layerId ][ n++ ] = j
NumValDlt[ layerId ] = n
```

I.7.4.3.3.8 Delta depth look-up table semantics

num_val_delta_dlt specifies the number of elements in the list deltaList. The length of num_val_delta_dlt syntax element is pps_bit_depth_for_depth_layers_minus8 + 8 bits.

max_diff specifies the maximum difference between two consecutive elements in the list deltaList. The length of max_diff syntax element is pps_bit_depth_for_depth_layers_minus8 + 8 bits. When not present, the value of max_diff is inferred to be equal to 0.

min_diff_minus1 specifies the minimum difference between two consecutive elements in the list deltaList. min_diff_minus1 shall be in the range of 0 to max_diff − 1, inclusive. The length of the min_diff_minus1 syntax element is Ceil(Log2(max_diff + 1)) bits. When not present, the value of min_diff_minus1 is inferred to be equal to (max_diff − 1).

The variable minDiff is set equal to (min_diff_minus1 + 1).

delta_dlt_val0 specifies the 0-th element in the list deltaList. The length of the delta_dlt_val0 syntax element is pps_bit_depth_for_depth_layers_minus8 + 8 bits.

delta_val_diff_minus_min[k] plus minDiff specifies the difference between the k-th element and the (k − 1)-th element in the list deltaList. The length of delta_val_diff_minus_min[k] syntax element is Ceil(Log2(max_diff − minDiff + 1)) bits. When not present, the value of delta_val_diff_minus_min[k] is inferred to be equal to 0.

The list deltaList is derived as follows:

```
deltaList[ 0 ] = delta_dlt_val0
for( k = 1; k < num_val_delta_dlt; k++ )                                          (I-15)
    deltaList[ k ] = deltaList[ k − 1 ] + delta_val_diff_minus_min[ k ] + minDiff
```

The variables layerId and refLayerId are set equal to DepIdxToLId[i] and DepIdxToLId[0], respectively.

The variables DltVal[layerId][n] and NumValDlt[layerId] of the depth look-up table of the layer with nuh_layer_id equal to layerId are derived as follows:

```
for( n = 0, j = 0; j <= depthMaxValue; j++ ) {
    inRefDltFlag = 0
    if( dlt_pred_flag[ i ] )
        for( k = 0; k < NumValDlt[ refLayerId ]; k++ )
            inRefDltFlag = inRefDltFlag | | ( DltVal[ refLayerId ][ k ] = = j )
    inUpdateDltFlag = 0                                                           (I-16)
    for( k = 0; k < num_val_delta_dlt; k++ )
        inUpdateDltFlag = inUpdateDltFlag | | ( deltaList[ k ] = = j )
    if( inRefDltFlag != inUpdateDltFlag )
        DltVal[ layerId ][ n++ ] = j
}
NumValDlt[ layerId ] = n
```

I.7.4.3.4 Supplemental enhancement information RBSP semantics

The specifications in clause F.7.4.3.4 apply.

I.7.4.3.5 Access unit delimiter RBSP semantics

The specifications in clause F.7.4.3.5 apply.

I.7.4.3.6 End of sequence RBSP semantics

The specifications in clause F.7.4.3.6 apply.

I.7.4.3.7 End of bitstream RBSP semantics

The specifications in clause F.7.4.3.7 apply.

I.7.4.3.8 Filler data RBSP semantics

The specifications in clause F.7.4.3.8 apply.

I.7.4.3.9 Slice segment layer RBSP semantics

The specifications in clause F.7.4.3.9 apply.

I.7.4.3.10 RBSP slice segment trailing bits semantics

The specifications in clause F.7.4.3.10 apply.

I.7.4.3.11 RBSP trailing bits semantics

The specifications in clause F.7.4.3.11 apply.

I.7.4.3.12 Byte alignment semantics

The specifications in clause F.7.4.3.12 apply.

I.7.4.4 Profile, tier and level semantics

The specifications in clause F.7.4.4 apply.

I.7.4.5 Scaling list data semantics

The specifications in clause F.7.4.5 apply.

I.7.4.6 Supplemental enhancement information message semantics

The specifications in clause F.7.4.6 apply.

I.7.4.7 Slice segment header semantics

I.7.4.7.1 General slice segment header semantics

The specifications in clause F.7.4.7.1 apply with the following modifications and additions.

The variable DepthFlag is set equal to DepthLayerFlag[nuh_layer_id] and the variable ViewIdx is set equal to ViewOrderIdx[nuh_layer_id].

The list curCmpLIds and the variable numCurCmpLIds are derived as follows:

curCmpLIds = DepthFlag ? { nuh_layer_id } : RefPicLayerId

numCurCmpLIds = DepthFlag ? 1 : NumActiveRefLayerPics

The list inCmpRefViewIdcs[i], the variable cpAvailableFlag, and the variable allRefCmpLayersAvailFlag are derived as follows:

- The variables cpAvailableFlag and allRefCmpLayersAvailFlag are set equal to 1.
- For i in the range of 0 to numCurCmpLIds − 1, inclusive, the following applies:
  - The variable inCmpRefViewIdcs[i] is set equal to ViewOrderIdx[curCmpLIds[i]].
  - When CpPresentFlag[ViewIdx][inCmpRefViewIdcs[i]] is equal to 0, cpAvailableFlag is set equal to 0.
  - The variable refCmpCurLIdAvailFlag is set equal to 0.
  - When ViewCompLayerPresentFlag[inCmpRefViewIdcs[i]][!DepthFlag] is equal to 1, the following applies:
    - The variable j is set equal to LayerIdxInVps[ViewCompLayerId[inCmpRefViewIdcs[i]][!DepthFlag]].
    - When all of the following conditions are true, refCmpCurLIdAvailFlag is set equal to 1:
      - direct_dependency_flag[LayerIdxInVps[nuh_layer_id]][j] is equal to 1.
      - sub_layers_vps_max_minus1[j] is greater than or equal to TemporalId.
      - TemporalId is equal to 0 or max_tid_il_ref_pics_plus1[j][LayerIdxInVps[nuh_layer_id]] is greater than TemporalId.
  - When refCmpCurLIdAvailFlag is equal to 0, allRefCmpLayersAvailFlag is set equal to 0.

The variable inCmpPredAvailFlag is derived as follows:

- If allRefCmpLayersAvailFlag is equal to 0, inCmpPredAvailFlag is set equal to 0.
- Otherwise (allRefCmpLayersAvailFlag) is equal to 1, the following applies:
  - If DepthFlag is equal to 0, the following applies:

$$\text{inCmpPredAvailFlag} = \text{vsp\_mc\_enabled\_flag[DepthFlag]} \;||\; \text{dbbp\_enabled\_flag[DepthFlag]} \;||\; \text{depth\_ref\_enabled\_flag[DepthFlag]} \quad \text{(I-17)}$$

  - Otherwise (DepthFlag is equal to 1), the following applies:

$$\text{inCmpPredAvailFlag} = \text{intra\_contour\_enabled\_flag[DepthFlag]} \;||\; \text{cqt\_cu\_part\_pred\_enabled\_flag[DepthFlag]} \;||\; \text{tex\_mc\_enabled\_flag[DepthFlag]} \quad \text{(I-18)}$$

in_comp_pred_flag equal to 0 specifies that reference pictures required for inter-component prediction of the current picture may not be present and that inter-component prediction of the current picture is disabled. in_comp_pred_flag equal to 1 specifies all reference pictures required for inter-component prediction of the current picture are present and that inter-component prediction of the current picture is enabled. When not present, the value of in_comp_pred_flag is inferred to be equal to 0.

When in_comp_pred_flag is equal to 1, the following applies for i in the range of 0 to numCurCmpLIds − 1, inclusive:

– It is a requirement of bitstream conformance that there is a picture in the DPB with PicOrderCntVal equal to the PicOrderCntVal of the current picture, and a nuh_layer_id value equal to ViewCompLayerId[inCmpRefViewIdcs[i]][!DepthFlag].

The variables IvDiMcEnabledFlag, IvMvScalEnabledFlag, IvResPredEnabledFlag, VspMcEnabledFlag, DbbpEnabledFlag, DepthRefEnabledFlag, TexMcEnabledFlag, IntraContourEnabledFlag, IntraDcOnlyWedgeEnabledFlag, CqtCuPartPredEnabledFlag, InterDcOnlyEnabledFlag, SkipIntraEnabledFlag and DisparityDerivationFlag are derived as follows:

```
IvDiMcEnabledFlag = NumRefListLayers[ nuh_layer_id ] > 0  &&  iv_di_mc_enabled_flag[ DepthFlag ]        (I-19)

IvMvScalEnabledFlag = iv_mv_scal_enabled_flag[ DepthFlag ]  &&  ViewIdx  !=  0                           (I-20)

IvResPredEnabledFlag = NumRefListLayers[ nuh_layer_id ] > 0
                                       &&  iv_res_pred_enabled_flag[ DepthFlag ]                         (I-21)

VspMcEnabledFlag = NumRefListLayers[ nuh_layer_id ] > 0  &&
              vsp_mc_enabled_flag[ DepthFlag ]  &&  in_comp_pred_flag  &&  cpAvailableFlag               (I-22)

DbbpEnabledFlag = dbbp_enabled_flag[ DepthFlag ]  &&  in_comp_pred_flag                                   (I-23)

DepthRefEnabledFlag = depth_ref_enabled_flag[ DepthFlag ]  &&  in_comp_pred_flag
                                                                    &&  cpAvailableFlag                  (I-24)

TexMcEnabledFlag = tex_mc_enabled_flag[ DepthFlag ]  &&  in_comp_pred_flag                                (I-25)

IntraContourEnabledFlag = intra_contour_enabled_flag[ DepthFlag ]  &&  in_comp_pred_flag                  (I-26)

IntraDcOnlyWedgeEnabledFlag = intra_dc_only_wedge_enabled_flag[ DepthFlag ]                               (I-27)

CqtCuPartPredEnabledFlag = cqt_cu_part_pred_enabled_flag[ DepthFlag ]  &&  in_comp_pred_flag  &&
    slice_type  !=  I  &&  !( nal_unit_type  >=  BLA_W_LP  &&  nal_unit_type  <=  RSV_IRAP_VCL23 )        (I-28)

InterDcOnlyEnabledFlag = inter_dc_only_enabled_flag[ DepthFlag ]                                          (I-29)

SkipIntraEnabledFlag = skip_intra_enabled_flag[ DepthFlag ]                                               (I-30)

DisparityDerivationFlag = IvDiMcEnabledFlag | |  IvResPredEnabledFlag  | |
                                             VspMcEnabledFlag  | | DbbpEnabledFlag                       (I-31)
```

When TexMcEnabledFlag is equal to 1, or CqtCuPartPredEnabledFlag is equal to 1, or IntraContourEnabledFlag is equal to 1, let TexturePic be the picture in the current access unit with nuh_layer_id equal to ViewCompLayerId[ViewIdx][0].

num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the num_inter_layer_ref_pics_minus1 syntax element is Ceil(Log2(NumRefListLayers[nuh_layer_id])) bits. The value of num_inter_layer_ref_pics_minus1 shall be in the range of 0 to NumRefListLayers[nuh_layer_id] − 1, inclusive.

The variables numRefLayerPics and refLayerPicIdc[j] are derived as follows:

```
for( i = 0, j = 0; i < NumRefListLayers[ nuh_layer_id ]; i++ ) {
    refLayerIdx = LayerIdxInVps[ IdRefListLayer[ nuh_layer_id ][ i ] ]
    if( sub_layers_vps_max_minus1[ refLayerIdx ]  >=  TemporalId  &&  ( TemporalId  = =  0  | |     (I-32)
            max_tid_il_ref_pics_plus1[ refLayerIdx ][ LayerIdxInVps[ nuh_layer_id ] ] > TemporalId ) )
        refLayerPicIdc[ j++ ] = i
}
numRefLayerPics = j
```

The variable NumActiveRefLayerPics is derived as follows:

```
if( nuh_layer_id  = =  0  | |  numRefLayerPics  = =  0 )
    NumActiveRefLayerPics = 0
else if( default_ref_layers_active_flag )
    NumActiveRefLayerPics = numRefLayerPics
else if( !inter_layer_pred_enabled_flag )                                                           (I-33)
    NumActiveRefLayerPics = 0
else if( max_one_active_ref_layer_flag  | |  NumRefListLayers[ nuh_layer_id ]  = =  1 )
    NumActiveRefLayerPics = 1
else
    NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1
```

All slices of a coded picture shall have the same value of NumActiveRefLayerPics.

inter_layer_pred_layer_idc[i] specifies the variable, RefPicLayerId[i], representing the nuh_layer_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the inter_layer_pred_layer_idc[i] syntax element is Ceil(Log2(NumRefListLayers[nuh_layer_id])) bits. The value of inter_layer_pred_layer_idc[i] shall be in the range of 0 to NumRefListLayers[nuh_layer_id] − 1, inclusive. When i is greater than 0, inter_layer_pred_layer_idc[i] shall be greater than inter_layer_pred_layer_idc[i − 1]. When not present, the value of inter_layer_pred_layer_idc[i] is inferred to be equal to refLayerPicIdc[i].

The variables RefPicLayerId[i] for all values of i in the range of 0 to NumActiveRefLayerPics − 1, inclusive, are derived as follows:

```
for( i = 0, j = 0; i < NumActiveRefLayerPics; i++ )                                    (I-34)
    RefPicLayerId[ i ] = IdRefListLayer[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]
```

The variable NumExtraMergeCand is derived as follows:

NumExtraMergeCand = IvDiMcEnabledFlag || TexMcEnabledFlag || VspMcEnabledFlag (I-35)

five_minus_max_num_merge_cand specifies the maximum number of merging motion vector prediction (MVP) candidates supported in the slice subtracted from (5 + NumExtraMergeCand).

The maximum number of merging MVP candidates, MaxNumMergeCand is derived as follows:

MaxNumMergeCand = 5 + NumExtraMergeCand − five_minus_max_num_merge_cand (I-36)

The value of MaxNumMergeCand shall be in the range of 1 to (5 + NumExtraMergeCand), inclusive.

slice_ic_enabled_flag equal to 1 specifies that the illu_comp_flag[x0][y0] syntax element may be present in coding units of the current slice. slice_ic_enabled_flag equal to 0 specifies that the illu_comp_flag[x0][y0] syntax element is not present in coding units of the current slice. When not present, the value of slice_ic_enabled_flag is inferred to be equal to 0.

slice_ic_disabled_merge_zero_idx_flag equal to 1 specifies that the illu_comp_flag[x0][y0] syntax element is not present in coding units of the current slice when merge_flag[x0][y0] is equal to 1 and merge_idx[x0][y0] is equal to 0. slice_ic_disabled_merge_zero_idx_flag equal to 0 specifies that illu_comp_flag[x0][y0] syntax element may be present in coding units of the current slice when merge_flag[x0][y0] is equal to 1 and merge_idx[x0][y0] is equal to 0. When not present, the value of slice_ic_disabled_merge_zero_idx_flag is inferred to be equal to 0.

cp_scale[j], **cp_off**[j], **cp_inv_scale_plus_scale**[j], and **cp_inv_off_plus_off**[j] specify parameters for the derivation of a horizontal component of a disparity vector from a depth value. When not present and CpPresentFlag[ViewIdx][j] is equal to 1, the values of cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] are inferred to be equal to vps_cp_scale[ViewIdx][j], vps_cp_off[ViewIdx][j], vps_cp_inv_scale_plus_scale[ViewIdx][j], and vps_cp_inv_off_plus_off[ViewIdx][j], respectively. It is a requirement of bitstream conformance, that the values of cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] in a slice header having a ViewIdx equal to viewIdxA and the values of cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] in a slice header having a ViewIdx equal to viewIdxB shall be the same, when viewIdxA is equal to viewIdxB.

The variable DepthToDisparityB[j][d] specifying the horizontal component of a disparity vector between the current view and the view with ViewIdx equal j corresponding to the depth value d in the view with ViewIdx equal to j and the variable DepthToDisparityF[j][d] specifying the horizontal component of a disparity vector between the view with ViewIdx equal j and the current view corresponding to the depth value d in the current view are derived as follows:

– The variable log2Div is set equal to (BitDepth_Y − 1 + cp_precision).

– For d in range of 0 to ((1 << BitDepth_Y) − 1), inclusive, the following applies:

  – For m in the range of 0 to (num_cp[ViewIdx] − 1), inclusive, the following applies:

j = cp_ref_voi[ViewIdx][m] (I-37)

offset = (cp_off[j] << BitDepth_Y) + ((1 << log2Div) >> 1) (I-38)

scale = cp_scale[j] (I-39)

DepthToDisparityB[j][d] = (scale * d + offset) >> log2Div (I-40)

invOffset = ((cp_inv_off_plus_off[j] − cp_off[j]) << BitDepth_Y) + ((1 << log2Div) >> 1) (I-41)

invScale = cp_inv_scale_plus_scale[j] − cp_scale[j] (I-42)

DepthToDisparityF[j][d] = (invScale * d + invOffset) >> log2Div (I-43)

I.7.4.7.2 Reference picture list modification semantics

The specifications in clause F.7.4.7.2 apply.

I.7.4.7.3 Weighted prediction parameters semantics

The specifications in clause F.7.4.7.3 apply.

I.7.4.8 Short-term reference picture set semantics

The specifications in clause F.7.4.8 apply.

I.7.4.9 Slice segment data semantics

I.7.4.9.1 General slice segment data semantics

The specifications in clause F.7.4.9.1 apply.

I.7.4.9.2 Coding tree unit semantics

The specifications in clause F.7.4.9.2 apply.

I.7.4.9.3 Sample adaptive offset semantics

The specifications in clause F.7.4.9.3 apply.

I.7.4.9.4 Coding quadtree semantics

The specifications in clause F.7.4.9.4 apply with the following modifications:

The variable predSplitCuFlag specifying whether the split_cu_flag[x0][y0] syntax element is inter-component predicted is derived as follows:

– If CqtCuPartPredEnabledFlag is equal to 1, the following applies:
  – Let colTextCu be the coding unit containing the luma coding block covering the luma location (x0, y0) in the picture TexturePic.
  – The variable log2TextCbSize is set equal to log2CbSize of the coding unit colTextCu.
  – The variable predSplitCuFlag is set equal to (log2CbSize <= log2TextCbSize).
– Otherwise (CqtCuPartPredEnabledFlag is equal to 0), predSplitCuFlag is set equal to 0.

split_cu_flag[x0][y0] specifies whether a coding unit is split into coding units with half horizontal and vertical size. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When split_cu_flag[x0][y0] is not present, the following applies:

– If log2CbSize is greater than MinCbLog2SizeY and predSplitCuFlag is equal to 0, the value of split_cu_flag[x0][y0] is inferred to be equal to 1.
– Otherwise (log2CbSize is equal to MinCbLog2SizeY or predSplitCuFlag is equal to 1), the value of split_cu_flag[x0][y0] is inferred to be equal to 0.

I.7.4.9.5 Coding unit semantics

The specifications in clause F.7.4.9.5 apply with the following modifications and additions:

cu_skip_flag[x0][y0] equal to 1 specifies that for the current coding unit, when decoding a P or B slice, no more syntax elements except the merging candidate index merge_idx[x0][y0], the iv_res_pred_weight_idx[x0][y0] syntax element, and the illu_comp_flag[x0][y0] syntax element may be parsed after cu_skip_flag[x0][y0]. cu_skip_flag[x0][y0] equal to 0 specifies that the coding unit is not skipped. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When cu_skip_flag[x0][y0] is not present, it is inferred to be equal to 0.

skip_intra_flag[x0][y0] equal to 1 specifies that for the current coding unit no more syntax elements except skip_intra_mode_idx[x0][y0] are parsed after skip_intra_flag[x0][y0]. skip_intra_flag[x0][y0] equal to 0 specifies that more syntax elements may be parsed after skip_intra_flag[x0][y0]. When not present, the value of skip_intra_flag[x0][y0] is inferred to be equal to 0.

pred_mode_flag equal to 0 specifies that the current coding unit is coded in inter prediction mode. pred_mode_flag equal to 1 specifies that the current coding unit is coded in intra prediction mode. The variable CuPredMode[x][y] is derived as follows for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1:

– If pred_mode_flag is equal to 0, CuPredMode[x][y] is set equal to MODE_INTER.

– Otherwise (pred_mode_flag is equal to 1), CuPredMode[x][y] is set equal to MODE_INTRA.

When pred_mode_flag is not present, the variable CuPredMode[x][y] is derived as follows for x = x0..x0 + nCbS − 1 and y = y0..y0 + nCbS − 1:

– If slice_type is equal to I or skip_intra_flag[x0][y0] is equal to 1, CuPredMode[x][y] is inferred to be equal to MODE_INTRA.

– Otherwise (slice_type is equal to P or B and skip_intra_flag[x0][y0] is equal to 0), when cu_skip_flag[x0][y0] is equal to 1, CuPredMode[x][y] is inferred to be equal to MODE_SKIP.

The variables predPartModeFlag and partPredIdc are derived as follows:

– If CqtCuPartPredEnabledFlag is equal to 1, the following applies:

  – Let colTextCu be the coding unit containing the luma coding block covering the luma location (x0, y0) in the picture TexturePic.

  – The variables log2TextCbSize and partTextMode are set equal to log2CbSize and PartMode, respectively, of the coding unit colTextCu.

  – The variable predPartModeFlag is derived as follows:

$$\text{predPartModeFlag} = \text{log2TextCbSize} == \text{log2CbSize} \ \&\& \ \text{partTextMode} == \text{PART\_2Nx2N} \quad \text{(I-44)}$$

  – The variable partPredIdc is derived as follows:

    – If one or more of the following conditions are true, partPredIdc is set equal to 0:

      – log2TextCbSize is not equal to log2CbSize.

      – partTextMode is equal to PART_2Nx2N or PART_NxN.

    – Otherwise, if partTextMode is equal to PART_2NxN, PART_2NxnU, or PART_2NxnL, partPredIdc is set equal to 1.

    – Otherwise, partPredIdc is set equal to 2.

– Otherwise (CqtCuPartPredEnabledFlag is equal to 0), predPartModeFlag and partPredIdc are set equal to 0.

part_mode specifies partitioning mode of the current coding unit. The semantics of part_mode depend on CuPredMode[x0][y0]. The variables PartMode and IntraSplitFlag are derived from the value of part_mode and partPredIdc as defined in Table I.1

Table I.1 – Name association to prediction mode and partitioning type

CuPredMode[x0][y0]	part_mode	partPredIdc	IntraSplitFlag	PartMode

MODE_INTRA	0	0	0	PART_2Nx2N
	1	0	1	PART_NxN
MODE_INTER	0	0	0	PART_2Nx2N
	1	0	0	PART_2NxN
	2	0	0	PART_Nx2N
	3	0	0	PART_NxN
	4	0	0	PART_2NxnU
	5	0	0	PART_2NxnD
	6	0	0	PART_nLx2N
	7	0	0	PART_nRx2N
	0	1	0	PART_2Nx2N
	1	1	0	PART_2NxN
	2	1	0	PART_2NxnU
	3	1	0	PART_2NxnD
	0	2	0	PART_2Nx2N
	1	2	0	PART_Nx2N
	2	2	0	PART_nLx2N
	3	2	0	PART_nRx2N

The value of part_mode is restricted as follows:

– If CuPredMode[x0][y0] is equal to MODE_INTRA, part_mode shall be equal to 0 or 1.

– Otherwise (CuPredMode[x0][y0] is equal to MODE_INTER), the following applies:

  – If partPredIdc is equal to 0, the following applies:

    – If log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 1, part_mode shall be in the range of 0 to 2, inclusive, or in the range of 4 to 7, inclusive.

    – Otherwise, if log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 0, or log2CbSize is equal to 3, part_mode shall be in the range of 0 to 2, inclusive.

    – Otherwise (log2CbSize is greater than 3 and less than or equal to MinCbLog2SizeY), the value of part_mode shall be in the range of 0 to 3, inclusive.

  – Otherwise (partPredIdc is not equal to 0), the following applies:

    – If log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 1, part_mode shall be in the range of 0 to 3, inclusive.

    – Otherwise (log2CbSize is equal to MinCbLog2SizeY or amp_enabled_flag is equal to 0), part_mode shall be in the range of 0 to 1, inclusive.

When part_mode is not present, the variables PartMode and IntraSplitFlag are derived as follows:

– PartMode is set equal to PART_2Nx2N.

– IntraSplitFlag is set equal to 0.

rqt_root_cbf equal to 1 specifies that the transform_tree( ) syntax structure is present for the current coding unit. rqt_root_cbf equal to 0 specifies that the transform_tree( ) syntax structure is not present for the current coding unit. When not present, the value of rqt_root_cbf is inferred to be equal to !DcOnlyFlag[x0][y0].

I.7.4.9.5.1 Intra mode extension semantics

no_dim_flag[x0][y0] equal to 1 specifies that the intra modes INTRA_WEDGE or INTRA_CONTOUR are not used for the current prediction unit. no_dim_flag[x0][y0] equal to 0 specifies that the intra mode INTRA_WEDGE or INTRA_CONTOUR is used for the current prediction unit. When not present, the value of no_dim_flag[x0][y0] is inferred to be equal to 1. When log2CbSize is greater than MaxTbLog2SizeY and DcOnlyFlag[x0][y0] is equal to 0, the value of no_dim_flag[x0][y0] shall be equal to 1.

For x = x0..x0 + (1 << log2PbSize) − 1, y = y0..y0 + (1 << log2PbSize) − 1, the variable DimFlag[x][y] is derived as follows:

DimFlag[x][y] = !no_dim_flag[x0][y0] (I-45)

depth_intra_mode_idx_flag[x0][y0] equal to 0, when DimFlag[x0][y0] is equal to 1, specifies that the intra mode INTRA_WEDGE is used for the current prediction unit. depth_intra_mode_idx_flag[x0][y0] equal to 1, when DimFlag[x0][y0] is equal to 1, specifies that the intra mode INTRA_CONTOUR is used for the current prediction unit. When not present, the value of depth_intra_mode_idx_flag[x0][y0] is inferred to be equal to (!IntraDcOnlyWedgeEnabledFlag || IntraContourEnabledFlag). When DimFlag[x0][y0] is equal to 1 and nal_unit_type is equal to BLA_W_LP, BLA_W_RADL, BLA_N_LP, IDR_W_RADL or IDR_N_LP, it is a requirement of bitstream conformance that depth_intra_mode_idx_flag[x0][y0] is equal to 0.

wedge_full_tab_idx[x0][y0] specifies the index of the partition pattern in the list WedgePatternTable[log2PbSize] when the intra mode INTRA_WEDGE is used for the current prediction unit.

I.7.4.9.5.2 Coding unit extension semantics

skip_intra_mode_idx[x0][y0] equal to 0, when skip_intra_flag[x0][y0] is equal to 1, specifies that the intra prediction mode INTRA_ANGULAR26 is used for the current prediction unit. skip_intra_mode_idx[x0][y0] equal to 1, when skip_intra_flag[x0][y0] is equal to 1, specifies that the intra prediction mode INTRA_ANGULAR10 is used for the current prediction unit. skip_intra_mode_idx[x0][y0] equal to 2 or 3, when skip_intra_flag[x0][y0] is equal to 1, specifies that the intra prediction mode INTRA_SINGLE is used for the current prediction unit.

dbbp_flag[x0][y0] equal to 1 specifies that the decoding process for inter sample prediction for depth predicted sub-block partitions is used for the current coding unit. dbbp_flag[x0][y0] equal to 0 specifies that the decoding process for inter sample prediction for depth predicted sub-block partitions is not used for the current coding unit. When not present, the value of dbbp_flag[x0][y0] is inferred to be equal to 0.

For x = x0..x0 + (1 << log2CbSize) − 1, y = y0..y0 + (1 << log2CbSize) − 1, the variable DbbpFlag[x][y] is derived as follows:

DbbpFlag[x][y] = dbbp_flag[x0][y0] (I-46)

dc_only_flag[x0][y0] equal to 1 specifies that the transform_tree( ) syntax structure is not present for the current coding unit and that the depth_dcs( ) syntax structure is present for the current coding unit. dc_only_flag[x0][y0] equal to 0 specifies the transform_tree( ) syntax structure may be present for the current coding unit and that the depth_dcs( ) syntax structure may be present for the current coding unit. When not present, the value of dc_only_flag[x0][y0] is inferred to be equal to 0. It is a requirement of bitstream conformance, that when pcm_flag[x0][y0] is equal to 1, the value of dc_only_flag[x0][y0] shall be equal to 0.

For x = x0..x0 + (1 << log2CbSize) − 1, y = y0..y0 + (1 << log2CbSize) − 1, the variable DcOnlyFlag[x][y] is derived as follows:

DcOnlyFlag[x][y] = dc_only_flag[x0][y0] (I-47)

iv_res_pred_weight_idx[x0][y0] not equal to 0 specifies that the bilinear sample interpolation and residual prediction process is used for the current coding unit and the index of the weighting factor used in the bilinear sample interpolation and residual prediction process. iv_res_pred_weight_idx[x0][y0] equal to 0 specifies that the bilinear sample interpolation and residual prediction process is not used for the current coding unit. When not present, the value of iv_res_pred_weight_idx[x0][y0] is inferred to be equal to 0.

When CuPredMode[x0][y0] is not equal to MODE_INTRA, the variable icCuEnableFlag is derived as follows:

– If merge_flag[x0][y0] is equal to 1, the following applies:

  icCuEnableFlag = (merge_idx[x0][y0] != 0) || !slice_ic_disabled_merge_zero_idx_flag (I-48)

– Otherwise (merge_flag[x0][y0] is equal to 0), the following applies:

  – For X in the range of 0 to 1, inclusive, the variable refViewIdxLX is set equal to ViewIdx(RefPicListX[ref_idx_lX[x0][y0]]).

  – The flag icCuEnableFlag is derived as follows:

    icCuEnableFlag = (inter_pred_idc[x0][y0] != Pred_L0 && refViewIdxL1 != ViewIdx) ||
    (inter_pred_idc[x0][y0] != Pred_L1 && refViewIdxL0 != ViewIdx) (I-49)

illu_comp_flag[x0][y0] equal to 1 specifies that the illumination compensated sample prediction process is used for the current coding unit. illu_comp_flag[x0][y0] equal to 0 specifies that the illumination compensated sample prediction process is not used for the current coding unit. When not present, the value of illu_comp_flag[x0][y0] is inferred to be equal to 0.

For x = x0..x0 + (1 << log2CbSize) − 1, y = y0..y0 + (1 << log2CbSize) − 1, the variable IlluCompFlag[x][y] is

derived as follows:

IlluCompFlag[x][y] = illu_comp_flag[x0][y0] (I-50)

I.7.4.9.5.3 Depth DCs semantics

depth_dc_present_flag[x0 + k][y0 + j] equal to 1 specifies that the depth_dc_abs[x0 + k][y0 + j][i] syntax element is present and that the depth_dc_sign_flag[x0 + k][y0 + j][i] syntax element may be present. depth_dc_present_flag[x0 + k][y0 + j] equal to 0 specifies that the depth_dc_abs[x0 + k][y0 + j][i] and depth_dc_sign_flag[x0 + k][y0 + j][i] syntax elements are not present. When not present, the value of depth_dc_present_flag[x0 + k][y0 + j] is inferred to be equal to 1.

depth_dc_abs[x0 + k][y0 + j][i] and **depth_dc_sign_flag**[x0 + k][y0 + j][i] are used to derive DcOffset[x0 + k][y0 + j][i]. When not present, the values of depth_dc_abs[x0 + k][y0 + j][i] and depth_dc_sign_flag[x0 + k][y0 + j][i] are inferred to be equal to 0. The variable DcOffset[x0 + k][y0 + j][i] is derived as follows:

DcOffset[x0 + k][y0 + j][i] = (1 − 2 * depth_dc_sign_flag[x0 + k][y0 + j][i]) *
(depth_dc_abs[x0 + k][y0 + j][i] − dcNumSeg + 2) (I-51)

I.7.4.9.6 Prediction unit semantics

The specifications in clause F.7.4.9.6 apply with the following addition at the end of the specification of semantics of inter_pred_idc[x0][y0]:

It is a requirement of bitstream conformance that, when DbbpFlag[x0][y0] is equal to 1, inter_pred_idc[x0][y0] shall not be equal to PRED_BI.

I.7.4.9.7 PCM sample semantics

The specifications in clause F.7.4.9.7 apply.

I.7.4.9.8 Transform tree semantics

The specifications in clause F.7.4.9.8 apply with the following additions at the end of the specification of split_transform_flag[x0][y0][trafoDepth]:

When DimFlag[x0][y0] is equal to 1 and PartMode is equal to PART_2Nx2N, the value of split_transform_flag[x0][y0][0] shall be equal to 0.

When DimFlag[x0][y0] is equal to 1 and PartMode is equal to PART_NxN, the value of split_transform_flag[x0][y0][1] shall be equal to 0.

I.7.4.9.9 Motion vector difference coding semantics

The specifications in clause F.7.4.9.9 apply.

I.7.4.9.10 Transform unit semantics

The specifications in clause F.7.4.9.10 apply.

I.7.4.9.11 Residual coding semantics

The specifications in clause F.7.4.9.11 apply.

I.7.4.9.12 Cross-component prediction semantics

The specifications in clause F.7.4.9.12 apply.

I.7.4.9.13 Palette mode semantics

The specifications in clause F.7.4.9.13 apply.

I.7.4.9.14 Delta QP semantics

The specifications in clause F.7.4.9.14 apply.

I.7.4.9.15 Chroma QP offset semantics

The specifications in clause F.7.4.9.15 apply.

I.8 Decoding process

I.8.1 General decoding process

I.8.1.1 General

The specifications in clause F.8.1.1 apply.

I.8.1.2 Decoding process for a coded picture with nuh_layer_id greater than 0

The decoding process for the current picture CurrPic is as follows:

1. The decoding of NAL units is specified in clause I.8.2.
2. The processes in clauses G.8.1.3, F.8.3.4 and I.8.3.1 to I.8.3.5 specify the following decoding processes using syntax elements in the slice segment layer and above:
   - Prior to decoding the first slice of the current picture, clause G.8.1.3 is invoked.
   - At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause F.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
   - When DisparityDerivationFlag is equal to 1 and DepthFlag is equal to 0, the derivation process for the candidate picture list for disparity vector derivation in clause I.8.3.1 is invoked at the beginning of the decoding process for each P or B slice.
   - The variable DefaultRefViewIdx is set equal to −1 and the variable DispAvailFlag is set equal to 0.
   - When DisparityDerivationFlag is equal to 1, the derivation process for the default reference view order index for disparity derivation as specified in clause I.8.3.2 is invoked at the beginning of the decoding process for each P or B slice.
   - When DltFlag[nuh_layer_id] is equal to 1, the derivation process for a depth look-up table in clause I.8.3.3 is invoked at the beginning of the decoding process of the first slice segment of a coded picture.
   - At the beginning of the decoding process for each P or B slice, the derivation process for the alternative target reference index for temporal motion vector prediction in merge mode as specified in clause I.8.3.4 is invoked.
   - When IvResPredEnabledFlag is equal to 1, the derivation process for the target reference index for residual prediction as specified in clause I.8.3.5 is invoked at the beginning of the decoding process for each P or B slice.
3. The processes in clauses I.8.4, I.8.5, I.8.6, and I.8.7, specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments, and the division of the slice segments into CTUs each form a partitioning of the picture.

I.8.2 NAL unit decoding process

The specifications in clause G.8.2 apply.

I.8.3 Slice decoding process

I.8.3.1 Derivation process for the candidate picture list for disparity vector derivation

The function tempId(picA) is specified as follows:

tempId(picA) = TemporalId of picA (I-52)

The variable NumDdvCandPics is set equal to 0 and when slice_temporal_mvp_enabled_flag is equal to 1, the list DdvCandPicList is constructed as follows:

1. The variable X is set equal to (1 − collocated_from_l0_flag), the variable DdvCandPicList[0] is set equal to RefPicListX[collocated_ref_idx], and NumDdvCandPics is set equal to 1.
2. The variables NumDdvCandPics, DdvCandPicList[1], and minTempIdRefs are derived as follows:

```
minTempIdRefs = 7
for( k = 0; k  <=  ( slice_type  = =  B ? 1 : 0 ); k++ ) {
    X = k ? collocated_from_l0_flag : ( 1 − collocated_from_l0_flag )
    for( i = 0; i  <=  num_ref_idx_lX_active_minus1; i++ )
```

```
        if( ViewIdx = = ViewIdx( RefPicListX[ i ] )
                && NumDdvCandPics != 2
                && ( X = = collocated_from_l0_flag | | i != collocated_ref_idx ) )          (I-53)
            if( RefPicListX[ i ] is an IRAP picture )
                DdvCandPicList[ NumDdvCandPics++ ] = RefPicListX[ i ]
            else
                minTempIdRefs = Min( minTempIdRefs, tempId( RefPicListX[ i ] ) )
}
```

3. When NumDdvCandPics is equal to 1, the following applies:

```
minPocDiff = 255
for( k = 0; k <= ( slice_type = = B ? 1 : 0 ); k++ ) {
    X = k ? collocated_from_l0_flag : ( 1 − collocated_from_l0_flag )
    for( i = 0; i <= num_ref_idx_lX_active_minus1; i++ )
        if( ViewIdx = = ViewIdx( RefPicListX[ i ] )
                && ( X = = collocated_from_l0_flag | | i != collocated_ref_idx )
                && tempId( RefPicListX[ i ] ) = = minTempIdRefs
                && Abs( DiffPicOrderCnt( CurrPic, RefPicListX[ i ] ) ) < minPocDiff ) {          (I-54)
            minPocDiff = Abs( DiffPicOrderCnt( CurrPic, RefPicListX[ i ] ) )
            Z = X
            candIdx = i
        }
}
if( minPocDiff < 255 )
    DdvCandPicList[ NumDdvCandPics++ ] = RefPicListZ[ candIdx ]
```

I.8.3.2 Derivation process for the default reference view order index for disparity derivation

This process is invoked when the current slice is a P or B slice.

The following applies for candViewIdx in the range of 0 to (ViewIdx − 1), inclusive:

– The following applies for X in the range of 0 to (slice_type = = B) ? 1 : 0, inclusive:
  – The following applies for i in the range of 0 to num_ref_idx_lX_active_minus1, inclusive:
    – When all of the following conditions are true, DefaultRefViewIdx is set equal to candViewIdx and DispAvailFlag is set equal to 1.
      – DispAvailFlag is equal to 0.
      – ViewIdx(RefPicListX[i]) is equal to candViewIdx.
      – DiffPicOrderCnt(CurrPic, RefPicListX[i]) is equal to 0.

When DepthFlag is equal to 1, DisparityDerivationFlag is equal to 1, and DispAvailFlag is equal to 1, it is a requirement of bitstream conformance that CpPresentFlag[ViewIdx][DefaultRefViewIdx] shall be equal to 1.

I.8.3.3 Derivation process for a depth look-up table

For i in the range of 0 to (1 << BitDepth_Y) − 1, inclusive, the list DltIdxToVal[i] specifying the depth value corresponding to the i-th index in the depth look-up table is derived as follows:

```
DltIdxToVal[ i ] = ( i < NumValDlt[ nuh_layer_id ] ) ? DltVal[ nuh_layer_id ][ i ] : 0          (I-55)
```

The list DltValToIdx[d] specifying the index in the depth look-up table corresponding to the depth value d is derived as follows:

```
for( d = 0; d < ( 1 << BitDepthY ); d++ ) {
    idxLower = 0
    foundFlag = 0
    for( iL = 1; iL < NumValDlt[ nuh_layer_id ]; iL++ )
        if( !foundFlag && DltIdxToVal[ iL ] > d ) {
            idxLower = iL − 1
            foundFlag = 1
        }                                                                                      (I-56)
    idxUpper = foundFlag ? ( idxLower + 1 ) : ( NumValDlt[ nuh_layer_id ] − 1 )
    if( Abs( d − DltIdxToVal[ idxLower ] ) < Abs ( d − DltIdxToVal[ idxUpper ] ) )
        DltValToIdx[ d ] = idxLower
```

```
        else
            DltValToIdx[ d ] = idxUpper
    }
```

I.8.3.4 Derivation process for the alternative target reference index for temporal motion vector prediction in merge mode

This process is invoked when the current slice is a P or B slice.

The variables AltRefIdxL0 and AltRefIdxL1 are set equal to −1 and the following applies for X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive:

```
zeroIdxLtFlag = RefPicListX[ 0 ] is a short-term reference picture ? 0 : 1
for( i = 1; i <= num_ref_idx_lX_active_minus1 && AltRefIdxLX == −1; i++ )
    if( ( zeroIdxLtFlag && RefPicListX[ i ] is a short-term reference picture ) ||                (I-57)
            ( !zeroIdxLtFlag && RefPicListX[ i ] is a long-term reference picture ) )
        AltRefIdxLX = i
```

I.8.3.5 Derivation process for the target reference index for residual prediction

This process is invoked when the current slice is a P or B slice.

For X in the range of 0 to 1, inclusive, the following applies:

- The variable RpRefIdxLX is set equal to −1 and the variable RpRefPicAvailFlagLX is set equal to 0.
- For i in the range of 0 to NumViews − 1, inclusive, the following applies:
  - The variable RefRpRefAvailFlagLX[ViewOIdxList[i]] is set equal to 0.

For X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive, the following applies:

- The variable minPocDiff is set equal to $2^{15} - 1$.
- For i in the range of 0 to num_ref_idx_lX_active_minus1, inclusive, the following applies:
  - The variable pocDiff is set equal to Abs(DiffPicOrderCnt(CurrPic, RefPicListX[i])).
  - When pocDiff is not equal to 0 and pocDiff is less than minPocDiff, the following applies:

    minPocDiff = pocDiff (I-58)

    RpRefIdxLX = i (I-59)

    RpRefPicAvailFlagLX = 1 (I-60)

- When DispAvailFlag is equal to 1 and RpRefPicAvailFlagLX is equal to 1, the following applies for i in the range of 0 to NumActiveRefLayerPics − 1, inclusive:
  - Let picV the picture in the current AU with nuh_layer_id equal to RefPicLayerId[i].
  - When there is a picture picA in one of the reference picture sets RefPicSetLtCurr, RefPicSetStCurrBefore, and RefPicSetStCurrAfter of picV with DiffPicOrderCnt(picA, RefPicListX[RpRefIdxLX]) equal to 0, RefRpRefAvailFlagLX[ViewIdx(RefPicLayerId[i])] is set equal to 1.

The variable RpRefPicAvailFlag is derived as follows:

RpRefPicAvailFlag = (RpRefPicAvailFlagL0 || RpRefPicAvailFlagL1) && DispAvailFlag (I-61)

When RpRefPicAvailFlag is equal to 1 and RefRpRefAvailFlagL0[ViewOIdxList[i]] is equal to 1 for any i in the range of 0 to NumViews − 1, inclusive, it is a requirement of bitstream conformance that PicOrderCnt(RefPicList0[RpRefIdxL0]) shall be the same for all slices of a coded picture.

When RpRefPicAvailFlag is equal to 1 and RefRpRefAvailFlagL1[ViewOIdxList[i]] is equal to 1 for any i in the range of 0 to NumViews − 1, inclusive, it is a requirement of bitstream conformance that PicOrderCnt(RefPicList1[RpRefIdxL1]) shall be the same for all slices of a coded picture.

I.8.4 Decoding process for coding units coded in intra prediction mode

I.8.4.1 General decoding process for coding units coded in intra prediction mode

The specifications in clause 8.4.1 apply with the following modification:

- All invocations of the process specified in clause 8.4.2 are replaced with invocations of the process specified in

clause I.8.4.2.

– All invocations of the process specified in clause 8.4.4.1 are replaced with invocations of the process specified in clause I.8.4.4.1.

I.8.4.2 Derivation process for luma intra prediction mode

Input to this process is a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

In this process, the luma intra prediction mode IntraPredModeY[xPb][yPb] is derived.

Table I.2 specifies the value for the intra prediction mode and the associated names.

Table I.2 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated name
0	INTRA_PLANAR
1	INTRA_DC
2..34	INTRA_ANGULAR2..INTRA_ANGULAR34
35	INTRA_WEDGE
36	INTRA_CONTOUR
37	INTRA_SINGLE

IntraPredModeY[xPb][yPb] labelled 0..34 represents directions of predictions as illustrated in Figure 8-1.

– If skip_intra_flag[xPb][yPb] is equal to 1, the following applies:
  – If skip_intra_mode_idx[xPb][yPb] is equal to 0, IntraPredModeY[xPb][yPb] is set equal to INTRA_ANGULAR26.
  – Otherwise, if skip_intra_mode_idx[xPb][yPb] is equal to 1, IntraPredModeY[xPb][yPb] is set equal to INTRA_ANGULAR10.
  – Otherwise (skip_intra_mode_idx[xPb][yPb] is equal to 2 or 3), IntraPredModeY[xPb][yPb] is set equal to INTRA_SINGLE.
– Otherwise, if DimFlag[xPb][yPb] is equal to 1, the following applies:
  – If depth_intra_mode_idx_flag[xPb][yPb] is equal to 0, IntraPredModeY[xPb][yPb] is set equal to INTRA_WEDGE.
  – Otherwise (depth_intra_mode_idx_flag[xPb][yPb] is equal to 1), IntraPredModeY[xPb][yPb] is set equal to INTRA_CONTOUR.
– Otherwise (skip_intra_flag[xPb][yPb] and DimFlag[xPb][yPb] are both equal to 0), IntraPredModeY[xPb][yPb] is derived by the following ordered steps:
  1. The neighbouring locations (xNbA, yNbA) and (xNbB, yNbB) are set equal to (xPb − 1, yPb) and (xPb, yPb − 1), respectively.
  2. For X being replaced by either A or B, the variables candIntraPredModeX are derived as follows:
     – The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (xPb, yPb) and the neighbouring location (xNbY, yNbY) set equal to (xNbX, yNbX) as inputs, and the output is assigned to availableX.
     – The candidate intra prediction mode candIntraPredModeX is derived as follows:
       – If availableX is equal to FALSE, candIntraPredModeX is set equal to INTRA_DC.
       – Otherwise, if CuPredMode[xNbX][yNbX] is not equal to MODE_INTRA or pcm_flag[xNbX][yNbX] is equal to 1, candIntraPredModeX is set equal to INTRA_DC,
       – Otherwise, if X is equal to B and yPb − 1 is less than ((yPb >> CtbLog2SizeY) << CtbLog2SizeY), candIntraPredModeB is set equal to INTRA_DC.
       – Otherwise, if IntraPredModeY[xNbX][yNbX] is greater than 34, candIntraPredModeX is set equal to INTRA_DC.

– Otherwise, candIntraPredModeX is set equal to IntraPredModeY[xNbX][yNbX].

3. The candModeList[x] with x = 0..2 is derived as follows:

– If candIntraPredModeB is equal to candIntraPredModeA, the following applies:

– If candIntraPredModeA is less than 2 (i.e., equal to INTRA_PLANAR or INTRA_DC), candModeList[x] with x = 0..2 is derived as follows:

candModeList[0] = INTRA_PLANAR (I-62)

candModeList[1] = INTRA_DC (I-63)

candModeList[2] = INTRA_ANGULAR26 (I-64)

– Otherwise, candModeList[x] with x = 0..2 is derived as follows:

candModeList[0] = candIntraPredModeA (I-65)

candModeList[1] = 2 + ((candIntraPredModeA + 29) % 32) (I-66)

candModeList[2] = 2 + ((candIntraPredModeA − 2 + 1) % 32) (I-67)

– Otherwise (candIntraPredModeB is not equal to candIntraPredModeA), the following applies:

– candModeList[0] and candModeList[1] are derived as follows:

candModeList[0] = candIntraPredModeA (I-68)

candModeList[1] = candIntraPredModeB (I-69)

– If neither of candModeList[0] and candModeList[1] is equal to INTRA_PLANAR, candModeList[2] is set equal to INTRA_PLANAR,

– Otherwise, if neither of candModeList[0] and candModeList[1] is equal to INTRA_DC, candModeList[2] is set equal to INTRA_DC,

– Otherwise, candModeList[2] is set equal to INTRA_ANGULAR26.

4. IntraPredModeY[xPb][yPb] is derived by applying the following procedure:

– If prev_intra_luma_pred_flag[xPb][yPb] is equal to 1, the IntraPredModeY[xPb][yPb] is set equal to candModeList[mpm_idx].

– Otherwise, IntraPredModeY[xPb][yPb] is derived by applying the following ordered steps:

1) The array candModeList[x], x = 0..2 is modified as the following ordered steps:

i. When candModeList[0] is greater than candModeList[1], both values are swapped as follows:

(candModeList[0], candModeList[1]) = Swap(candModeList[0], candModeList[1]) (I-70)

ii. When candModeList[0] is greater than candModeList[2], both values are swapped as follows:

(candModeList[0], candModeList[2]) = Swap(candModeList[0], candModeList[2]) (I-71)

iii. When candModeList[1] is greater than candModeList[2], both values are swapped as follows:

(candModeList[1], candModeList[2]) = Swap(candModeList[1], candModeList[2]) (I-72)

2) IntraPredModeY[xPb][yPb] is derived by the following ordered steps:

i. IntraPredModeY[xPb][yPb] is set equal to rem_intra_luma_pred_mode[xPb][yPb].

ii. For i equal to 0 to 2, inclusive, when IntraPredModeY[xPb][yPb] is greater than or equal to candModeList[i], the value of IntraPredModeY[xPb][yPb] is incremented by one.

I.8.4.3 Derivation process for chroma intra prediction mode

The specifications in clause 8.4.3 apply.

I.8.4.4 Decoding process for intra blocks

I.8.4.4.1 General decoding process for intra blocks

Inputs to this process are:

– a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
– a variable log2TrafoSize specifying the size of the current transform block,
– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,
– a variable predModeIntra specifying the intra prediction mode,
– a variable cIdx specifying the colour component of the current block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The luma sample location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture is derived as follows:

(xTbY, yTbY) = (cIdx = = 0) ? (xTb0, yTb0) : (xTb0 * SubWidthC, yTb0 * SubHeightC) (I-73)

The variable splitFlag is derived as follows:

– If DcOnlyFlag[xTbY][yTbY] is equal to 1, the following applies:

  splitFlag = !DimFlag[xTbY][yTbY] && (log2TrafoSize > MaxTbLog2SizeY) (I-74)

– Otherwise, if skip_intra_flag[xTbY][yTbY] is equal to 1, splitFlag is set equal to 0.
– Otherwise, if cIdx is equal to 0, splitFlag is set equal to split_transform_flag[xTbY][yTbY][trafoDepth].
– Otherwise, if all of the following conditions are true, splitFlag is set equal to 1.
  – cIdx is greater than 0
  – split_transform_flag[xTbY][yTbY][trafoDepth] is equal to 1
  – log2TrafoSize is greater than 2
– Otherwise, splitFlag is set equal to 0.

Depending on the value of splitFlag, the following applies:

– If splitFlag is equal to 1, the following ordered steps apply:
  1. The variables xTb1 and yTb1 are derived as follows:
     – If cIdx is equal to 0 or ChromaArrayType is not equal to 2, the following applies:
       – The variable xTb1 is set equal to xTb0 + (1 << (log2TrafoSize − 1)).
       – The variable yTb1 is set equal to yTb0 + (1 << (log2TrafoSize − 1)).
     – Otherwise (ChromaArrayType is equal to 2 and cIdx is greater than 0), the following applies:
       – The variable xTb1 is set equal to xTb0 + (1 << (log2TrafoSize − 1)).
       – The variable yTb1 is set equal to yTb0 + (2 << (log2TrafoSize − 1)).
  2. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb0, yTb0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
  3. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb1, yTb0), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
  4. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb0, yTb1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
  5. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb1, yTb1), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (splitFlag is equal to 0), for the variable blkIdx proceeding over the values 0..(cIdx > 0 && ChromaArrayType = = 2 ? 1 : 0), the following ordered steps apply:

1. The variable nTbS is set equal to 1 << log2TrafoSize.
2. The variable yTbOffset is set equal to blkIdx * nTbS.
3. The variable yTbOffsetY is set equal to yTbOffset * SubHeightC.
4. The variable residualDpcm is derived as follows:
   – If all of the following conditions are true, residualDpcm is set equal to 1.
     – implicit_rdpcm_enabled_flag is equal to 1.
     – either transform_skip_flag[xTbY][yTbY + yTbOffsetY][cIdx] is equal to 1, or cu_transquant_bypass_flag is equal to 1.
     – either predModeIntra is equal to 10, or predModeIntra is equal to 26.
   – Otherwise, residualDpcm is set equal to 0.
5. The general intra sample prediction process as specified in clause I.8.4.4.2.1 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the intra prediction mode predModeIntra, the transform block size nTbS, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.
6. The variable residualFlag is set equal to !(skip_intra_flag[xTb0][xTb0] | | DcOnlyFlag[xTb0][xTb0]) and depending on the value of residualFlag, the following applies:
   – If residualFlag is equal to 1, the following applies:
     – The scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location (xTbY, yTbY + yTbOffsetY), the variable trafoDepth, the variable cIdx, and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array resSamples.
     – When residualDpcm is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable mDir set equal to predModeIntra / 26, the variable nTbS, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
     – When cross_component_prediction_enabled_flag is equal to 1, ChromaArrayType is equal to 3, and cIdx is not equal to 0, the residual modification process for transform blocks using cross-component prediction as specified in clause 8.6.6 is invoked with the current luma transform block location (xTbY, yTbY), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array r_Y set equal to the corresponding luma residual sample array resSamples of the current transform block, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
   – Otherwise (residualFlag is equal to 0), for x, y = 0..nTbS − 1, resSamples[x][y] is set equal to 0.
7. The picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the variables nCurrSw and nCurrSh both set equal to nTbS, the variable cIdx, the (nTbS)x(nTbS) array predSamples, and the (nTbS)x(nTbS) array resSamples as inputs.

When trafoDepth is equal to 0, DcOnlyFlag[xTb0][yTb0] is equal to 1, and DimFlag[xTb0][yTb0] is equal to 0, the depth DC offset assignment process as specified in clause I.8.4.4.3 is invoked with the location (xTb0, yTb0), and the transform size trafoSize set equal to nTbS as inputs.

I.8.4.4.2 Intra sample prediction

I.8.4.4.2.1 General intra sample prediction

Inputs to this process are:

– a sample location (xTbCmp, yTbCmp) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
– a variable predModeIntra specifying the intra prediction mode,
– a variable nTbS specifying the transform block size,
– a variable cIdx specifying the colour component of the current block.

Outputs of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The nTbS * 4 + 1 neighbouring samples p[x][y] that are constructed samples prior to the deblocking filter process, with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1, are derived as follows:

– The neighbouring location (xNbCmp, yNbCmp) is specified by:

$$(xNbCmp, yNbCmp) = (xTbCmp + x, yTbCmp + y) \quad \text{(I-75)}$$

– The current luma location (xTbY, yTbY) and the neighbouring luma location (xNbY, yNbY) are derived as follows:

$$(xTbY, yTbY) = (cIdx == 0) ? (xTbCmp, yTbCmp) : (xTbCmp * SubWidthC, yTbCmp * SubHeightC) \quad \text{(I-76)}$$

$$(xNbY, yNbY) = (cIdx == 0) ? (xNbCmp, yNbCmp) : (xNbCmp * SubWidthC, yNbCmp * SubHeightC) \quad \text{(I-77)}$$

– The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the current luma location (xCurr, yCurr) set equal to (xTbY, yTbY) and the neighbouring luma location (xNbY, yNbY) as inputs, and the output is assigned to availableN.

– Each sample p[x][y] is derived as follows:

  – If one or more of the following conditions are true, the sample p[x][y] is marked as "not available for intra prediction":

    – The variable availableN is equal to FALSE.

    – CuPredMode[xNbY][yNbY] is not equal to MODE_INTRA and constrained_intra_pred_flag is equal to 1.

  – Otherwise, the sample p[x][y] is marked as "available for intra prediction" and the sample at the location (xNbCmp, yNbCmp) is assigned to p[x][y].

When at least one sample p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 is marked as "not available for intra prediction" and predModeIntra is not equal to INTRA_SINGLE, the reference sample substitution process for intra sample prediction in clause 8.4.4.2.2 is invoked with the samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1, nTbS, and cIdx as inputs, and the modified samples p[x][y] with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1 as output.

Depending on the value of predModeIntra, the following ordered steps apply:

1. When predModeIntra is in the range of 0 to 34, inclusive, intra_smoothing_disabled_flag is equal to 0, and either cIdx is equal to 0 or ChromaArrayType is equal to 3, the filtering process of neighbouring samples specified in clause 8.4.4.2.3 is invoked with the sample array p, the transform block size nTbS and the colour component index cIdx as inputs, and the output is reassigned to the sample array p.

2. The intra sample prediction process according to predModeIntra applies as follows:

   – If predModeIntra is equal to INTRA_PLANAR, the corresponding intra prediction mode specified in clause 8.4.4.2.4 is invoked with the sample array p and the transform block size nTbS as inputs, and the output is the predicted sample array predSamples.

   – Otherwise, if predModeIntra is equal to INTRA_DC, the corresponding intra prediction mode specified in clause 8.4.4.2.5 is invoked with the sample array p, the transform block size nTbS, and the colour component index cIdx as inputs, and the output is the predicted sample array predSamples.

   – Otherwise, if predModeIntra is in the range of INTRA_ANGULAR2..INTRA_ANGULAR34, the corresponding intra prediction mode specified in clause 8.4.4.2.6 is invoked with the intra prediction mode predModeIntra, the sample array p, the transform block size nTbS, and the colour component index cIdx as inputs, and the output is the predicted sample array predSamples.

   – Otherwise, if predModeIntra is equal to INTRA_WEDGE, the corresponding intra prediction mode specified in clause I.8.4.4.2.2 is invoked with the location (xTbY, yTbY), the sample array p, and the transform block size nTbS as inputs, and the output is the predicted sample array predSamples.

   – Otherwise, if predModeIntra is equal to INTRA_CONTOUR, the corresponding intra prediction mode specified in clause I.8.4.4.2.3 is invoked with the location (xTbY, yTbY), the sample array p, and the transform block size nTbS as inputs, and the output is the predicted sample array predSamples.

   – Otherwise, if predModeIntra is equal to INTRA_SINGLE, the corresponding intra prediction mode specified in clause I.8.4.4.2.4 is invoked with the location (xTbY, yTbY), the sample array p, and the transform block size nTbS as inputs, and the output is the predicted sample array predSamples.

I.8.4.4.2.2 Specification of intra prediction mode INTRA_WEDGE

Inputs to this process are:

– a luma location (xTb, yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,

– the neighbouring samples p[x][y], with x = −1, y = − 1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a variable nTbS specifying the transform block size.

Output of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The list WedgePatternTable and the variable NumWedgePattern are specified by the derivation process for a wedgelet partition pattern table in clause I.6.6.

The partition pattern wedgePattern[x][y] with x, y = 0..nTbS − 1 is derived as follows:

wedgePattern = WedgePatternTable[Log2(nTbS)][wedge_full_tab_idx[xTb][yTb]] (I-78)

The depth sub-block partition DC value derivation and assignment process as specified in clause I.8.4.4.2.5 is invoked with the neighbouring samples p[x][y], the partition pattern wedgePattern[x][y], the luma location (xTb, yTb), and the transform size nTbS as inputs, and the output is assigned to predSamples[x][y].

I.8.4.4.2.3 Specification of intra prediction mode INTRA_CONTOUR

Inputs to this process are:

– a luma location (xTb, yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a variable nTbS specifying the transform block size.

Output of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The partition pattern contourPattern[x][y], with x, y = 0..nTbS − 1 is derived as follows:

– The variable refSamples is set equal to the reconstructed picture sample array S_L of the picture TexturePic.

– The variable threshVal is derived as follows:

threshVal = (refSamples[xTb][yTb] + refSamples[xTb][yTb + nTbS − 1]
+ refSamples[xTb + nTbS − 1][yTb] + refSamples[xTb + nTbS − 1][yTb + nTbS − 1]) >> 2 (I-79)

– For x = 0..nTbS − 1 and y = 0..nTbS − 1, the following applies:

contourPattern[x][y] = (refSamples[xTb + x][yTb + y] > threshVal) (I-80)

The depth sub-block partition DC value derivation and assignment process as specified in clause I.8.4.4.2.5 is invoked with the neighbouring samples p[x][y], the partition pattern contourPattern[x][y], the luma location (xTb, yTb), and the transform size nTbS as inputs, and the output is assigned to predSamples[x][y].

I.8.4.4.2.4 Specification of intra prediction mode INTRA_SINGLE

Inputs to this process are:

– a luma location (xTb, yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a variable nTbS specifying the transform block size.

Output of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

Depending on the value of skip_intra_mode_idx[xTb][yTb], the luma location (xN, yN) is derived as follows:

– If skip_intra_mode_idx[xTb][yTb] is equal to 2, (xN, yN) is set equal to (−1, nTbS >> 1).

– Otherwise (skip_intra_mode_idx[xTb][yTb] is equal to 3), (xN, yN) is set equal to (nTbS >> 1, −1).

The variable singleSampleVal is derived as follows:

– If p[xN][yN] is marked as "available for intra prediction", singleSampleVal is set equal to p[xN][yN].

– Otherwise (p[xN][yN] is marked as "not available for intra prediction"), singleSampleVal is set equal to (1 << ($BitDepth_Y$ − 1))

The values of the prediction samples predSamples[x][y], with x, y = 0..nTbS − 1, are derived as follows:

predSamples[x][y] = singleSampleVal (I-81)

I.8.4.4.2.5 Depth sub-block partition DC value derivation and assignment process

Inputs to this process are:

– the neighbouring samples p[x][y], with x = −1, y = −1..nTbS * 2 − 1 and x = 0..nTbS * 2 − 1, y = −1,

– a partition pattern partitionPattern[x][y], with x, y = 0..nTbS − 1, specifying the sub-block partitions of the current prediction block,

– a luma location (xTb, yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,

– a variable nTbS specifying the transform block size.

Output of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS − 1.

The variables vertEdgeFlag and horEdgeFlag are derived as follows:

vertEdgeFlag = (partitionPattern[0][0] != partitionPattern[nTbS − 1][0]) (I-82)

horEdgeFlag = (partitionPattern[0][0] != partitionPattern[0][nTbS − 1]) (I-83)

The variables dcValBR and dcValLT are derived as follows:

– If vertEdgeFlag is equal to horEdgeFlag, the following applies:

  – The variable dcValBR is derived as follows:

    – If horEdgeFlag is equal to 1, the following applies:

      dcValBR = ((p[−1][nTbS − 1] + p[nTbS − 1][−1]) >> 1) (I-84)

    – Otherwise (horEdgeFlag is equal to 0), the following applies:

      vertAbsDiff = Abs(p[−1][0] − p[−1][nTbS * 2 − 1]) (I-85)

      horAbsDiff = Abs(p[0][−1] − p[nTbS * 2 − 1][−1]) (I-86)

      dcValBR = (horAbsDiff > vertAbsDiff) ? p[nTbS * 2 − 1][−1] : p[−1][nTbS * 2 − 1] (I-87)

  – The variable dcValLT is derived as follows:

    dcValLT = (p[−1][0] + p[0][−1]) >> 1 (I-88)

– Otherwise (horEdgeFlag is not equal to vertEdgeFlag), the following applies:

  dcValBR = horEdgeFlag ? p[−1][nTbS − 1] : p[nTbS − 1][−1] (I-89)

  dcValLT = horEdgeFlag ? p[(nTbS − 1) >> 1][−1] : p[−1][(nTbS − 1) >> 1] (I-90)

The predicted sample values predSamples[x][y] are derived as follows:

– For x in the range of 0 to (nTbS − 1), inclusive, the following applies:

  – For y in the range of 0 to (nTbS − 1), inclusive, the following applies:

    – The variables predDcVal and dcOffset are derived as follows:

      predDcVal = (partitionPattern[x][y] = = partitionPattern[0][0]) ? dcValLT : dcValBR (I-91)

      dcOffset = DcOffset[xTb][yTb][partitionPattern[x][y]] (I-92)

    – If DltFlag[nuh_layer_id] is equal to 0, the following applies:

      predSamples[x][y] = predDcVal + dcOffset (I-93)

    – Otherwise (DltFlag[nuh_layer_id] is equal to 1), the following applies:

      predSamples[x][y] = DltIdxToVal[$Clip1_Y$(DltValToIdx[predDcVal] + dcOffset)] (I-94)

I.8.4.4.3 Depth DC offset assignment process

Inputs to this process are:

– a luma location (xTb, yTb) specifying the top-left luma sample of the current transform block relative to the top-left luma sample of the current picture,

– a variable nTbS specifying the transform block size.

Output of this process is a modified reconstructed picture S_L before deblocking filtering.

Depending on the value of DltFlag[nuh_layer_id], the variable dcVal is derived as follows:

– If DltFlag[nuh_layer_id] is equal to 0, dcVal is set equal to DcOffset[xTb][yTb][0].

– Otherwise (DltFlag[nuh_layer_id] is equal to 1), dcVal is derived as follows:

$$\begin{aligned}\text{dcPred} = (\ & S_L[\ \text{xTb}\ ][\ \text{yTb}\ ] + S_L[\ \text{xTb}\ ][\ \text{yTb} + \text{nTbS} - 1\ ] + \\ & S_L[\ \text{xTb} + \text{nTbS} - 1\ ][\ \text{yTb}\ ] + S_L[\ \text{xTb} + \text{nTbS} - 1\ ][\ \text{yTb} + \text{nTbS} - 1\ ] + 2\ ) >> 2\end{aligned} \quad \text{(I-95)}$$

$$\text{dcVal} = \text{DltIdxToVal}[\ \text{Clip1}_Y(\ \text{DltValToIdx}[\ \text{dcPred}\ ] + \text{DcOffset}[\ \text{xTb}\ ][\ \text{yTb}\ ][\ 0\ ]\ )\ ] - \text{dcPred} \quad \text{(I-96)}$$

For x, y = 0..nTbS − 1, the variable S_L[x][y] is modified as follows:

$$S_L[\ \text{xTb} + \text{x}\ ][\ \text{yTb} + \text{y}\ ] = \text{Clip1}_Y(\ S_L[\ \text{xTb} + \text{x}\ ][\ \text{yTb} + \text{y}\ ] + \text{dcVal}\ ) \quad \text{(I-97)}$$

I.8.5 Decoding process for coding units coded in inter prediction mode

I.8.5.1 General decoding process for coding units coded in inter prediction mode

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the luma location (xCb, yCb) as input.

The variable $nCbS_L$ is set equal to 1 << log2CbSize. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to (1 << log2CbSize) / SubWidthC and the variable $nCbSh_C$ is set equal to (1 << log2CbSize) / SubHeightC.

The decoding process for coding units coded in inter prediction mode consists of following ordered steps:

1. When DisparityDerivationFlag is equal to 1, the following applies:
   – If DepthFlag is equal to 0, the derivation process for a disparity vector for texture layers as specified in clause I.8.5.5 is invoked with the luma location (xCb, yCb) and the coding block size $nCbS_L$ as inputs.
   – Otherwise (DepthFlag is equal to 1), the derivation process for a disparity vector for depth layers as specified in clause I.8.5.6 is invoked with the luma location (xCb, yCb) and the coding block size $nCbS_L$ as inputs.
2. The inter prediction process as specified in clause I.8.5.2 is invoked with the luma location (xCb, yCb) and the luma coding block size log2CbSize as inputs, and the output is the array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the arrays $predSamples_{Cb}$, and $predSamples_{Cr}$.
3. The decoding process for the residual signal of coding units coded in inter prediction mode specified in clause I.8.5.4 is invoked with the luma location (xCb, yCb) and the luma coding block size log2CbSize as inputs, and the outputs are the array $resSamples_L$ and, when ChromaArrayType is not equal to 0, the arrays $resSamples_{Cb}$, and $resSamples_{Cr}$.
4. The reconstructed samples of the current coding unit are derived as follows:
   – The picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the luma coding block location (xCb, yCb), the variable nCurrSw set equal to $nCbS_L$, the variable nCurrSh set equal to $nCbS_L$, the variable cIdx set equal to 0, the ($nCbS_L$)x($nCbS_L$) array predSamples set equal to $predSamples_L$, and the ($nCbS_L$)x($nCbS_L$) array resSamples set equal to $resSamples_L$ as inputs.
   – When ChromaArrayType is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location (xCb / SubWidthC, yCb / SubHeightC), the variable nCurrSw set equal to $nCbSw_C$, the variable nCurrSh

set equal to $nCbSh_C$, the variable cIdx set equal to 1, the $(nCbSw_C)x(nCbSh_C)$ array predSamples set equal to $predSamples_{Cb}$, and the $(nCbSw_C)x(nCbSh_C)$ array resSamples set equal to $resSamples_{Cb}$ as inputs.

– When ChromaArrayType is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location (xCb / SubWidthC, yCb / SubHeightC), the variable nCurrSw set equal to $nCbSw_C$, the variable nCurrSh set equal to $nCbSh_C$, the variable cIdx set equal to 2, the $(nCbSw_C)x(nCbSh_C)$ array predSamples set equal to $predSamples_{Cr}$, and the $(nCbSw_C)x(nCbSh_C)$ array resSamples set equal to $resSamples_{Cr}$ as inputs.

I.8.5.2 Inter prediction process

The specifications in clause 8.5.2 apply with the following modification:

– All invocations of the process specified in clause 8.5.3 are replaced with invocations of the process specified in clause I.8.5.3.

I.8.5.3 Decoding process for prediction units in inter prediction mode

I.8.5.3.1 General

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– a variable nPbW specifying the width of the current luma prediction block,

– a variable nPbH specifying the height of the current luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ of luma prediction samples, where $nCbS_L$ is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to nCbS. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to nCbS / SubWidthC and the variable $nCbSh_C$ is set equal to nCbS / SubHeightC.

The decoding process for prediction units in inter prediction mode consists of the following ordered steps:

1. The derivation process for motion vector components and reference indices as specified in clause I.8.5.3.2 is invoked with the luma coding block location (xCb, yCb), the luma prediction block location (xBl, yBl), the luma coding block size block nCbS, the luma prediction block width nPbW, the luma prediction block height nPbH, and the prediction unit index partIdx as inputs, and the luma motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, the prediction list utilization flags predFlagL0 and predFlagL1, and the flag subPbMotionFlag as outputs.

2. Depending on the value of subPbMotionFlag and DbbpFlag[xCb][yCb], the following applies:

   – If both subPbMotionFlag and DbbpFlag[xCb][yCb] are equal to 0, the decoding process for inter sample prediction as specified in clause I.8.5.3.3.1 is invoked with the luma coding block location (xCb, yCb), the luma prediction block location (xBl, yBl), the luma coding block size block nCbS, the luma prediction block width nPbW, the luma prediction block height nPbH, the luma motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1 as inputs, and the inter prediction samples (predSamples) that are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cr}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr, as outputs.

   – Otherwise, if subPbMotionFlag is equal to 1, the decoding process for inter sample prediction for rectangular

sub-block partitions as specified in clause I.8.5.3.4 is invoked with the luma coding block location (xCb, yCb), the luma prediction block location (xBl, yBl), the luma coding block size block nCbS, the luma prediction block width nPbW, and the luma prediction block height nPbH as inputs, and the inter prediction samples that are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of chroma prediction samples, one for each of the chroma components Cb and Cr, as outputs.

– Otherwise (DbbpFlag[xCb][yCb] is equal to 1), the decoding process for inter sample prediction for depth predicted sub-block partitions as specified in clause I.8.5.3.5 is invoked with the luma coding block location (xCb, yCb), the luma coding block size nCbS, the luma motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, the prediction list utilization flags predFlagL0 and predFlagL1, and the variable partIdx as inputs, and the inter prediction samples that are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of chroma prediction samples, one for each of the chroma components Cb and Cr, as outputs.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xBl..xBl + nPbW − 1 and y = yBl..yBl + nPbH − 1:

MvL0[xCb + x][yCb + y] = subPbMotionFlag ? SubPbArrayMvL0[xCb + x][yCb + y] : mvL0 (I-98)

MvL1[xCb + x][yCb + y] = subPbMotionFlag ? SubPbArrayMvL1[xCb + x][yCb + y] : mvL1 (I-99)

RefIdxL0[xCb + x][yCb + y] = subPbMotionFlag ?
SubPbArrayRefIdxL0[xCb + x][yCb + y] : refIdxL0 (I-100)

RefIdxL1[xCb + x][yCb + y] = subPbMotionFlag ?
SubPbArrayRefIdxL1[xCb + x][yCb + y] : refIdxL1 (I-101)

PredFlagL0[xCb + x][yCb + y] = subPbMotionFlag ?
SubPbArrayPredFlagL0[xCb + x][yCb + y] : predFlagL0 (I-102)

PredFlagL1[xCb + x][yCb + y] = subPbMotionFlag ?
SubPbArrayPredFlagL1[xCb + x][yCb + y] : predFlagL1 (I-103)

I.8.5.3.2 Derivation process for motion vector components and reference indices

I.8.5.3.2.1 General

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) of the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– the luma motion vectors mvL0 and mvL1,

– when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1,

– the flag subPbMotionFlag specifying, whether specifying whether the current PU is coded using sub-block partitions.

Let (xPb, yPb) specify the top-left sample location of the current luma prediction block relative to the top-left luma sample of the current picture where xPb = xCb + xBl and yPb = yCb + yBl.

Let the variables currPic and LX be the current picture and RefPicListX, with X being 0 or 1, of the current picture, respectively.

The function LongTermRefPic(aPic, aPb, refIdx, LX), with X being 0 or 1, is defined as follows:

– If the picture with index refIdx from reference picture list LX of the slice containing prediction block aPb in the picture aPic was marked as "used for long-term reference" at the time when aPic was the current picture, LongTermRefPic(aPic, aPb, refIdx, LX) is equal to 1.

– Otherwise, LongTermRefPic(aPic, aPb, refIdx, LX) is equal to 0.

The variables vspMcFlag, ivMcFlag, and subPbMotionFlag are set equal to 0.

For the derivation of the variables mvL0 and mvL1, refIdxL0 and refIdxL1, as well as predFlagL0 and predFlagL1, the following applies:

– If merge_flag[xPb][yPb] is equal to 1, the derivation process for luma motion vectors for merge mode as specified in clause I.8.5.3.2.7 is invoked with the luma location (xCb, yCb), the luma location (xPb, yPb), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, the prediction list utilization flags predFlagL0 and predFlagL1, the flag ivMcFlag, the flag vspMcFlag, and the flag subPbMotionFlag.

– Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, and refIdxLX, in PRED_LX, and in the syntax elements ref_idx_lX and MvdLX, the following applies:

1. The variables refIdxLX and predFlagLX are derived as follows:

   – If inter_pred_idc[xPb][yPb] is equal to PRED_LX or PRED_BI,

   $$\text{refIdxLX} = \text{ref\_idx\_lX[xPb][yPb]} \quad \text{(I-104)}$$

   $$\text{predFlagLX} = 1 \quad \text{(I-105)}$$

   – Otherwise, the variables refIdxLX and predFlagLX are specified by:

   $$\text{refIdxLX} = -1 \quad \text{(I-106)}$$

   $$\text{predFlagLX} = 0 \quad \text{(I-107)}$$

2. The variable mvdLX is derived as follows:

   $$\text{mvdLX[0]} = \text{MvdLX[xPb][yPb][0]} \quad \text{(I-108)}$$

   $$\text{mvdLX[1]} = \text{MvdLX[xPb][yPb][1]} \quad \text{(I-109)}$$

3. When predFlagLX is equal to 1, the derivation process for luma motion vector prediction in clause I.8.5.3.2.3 is invoked with the luma coding block location (xCb, yCb), the coding block size nCbS, the luma prediction block location (xPb, yPb), the variables nPbW, nPbH, refIdxLX, and the partition index partIdx as inputs, and the output being mvpLX.

4. When predFlagLX is equal to 1, the luma motion vector mvLX is derived as follows:

   $$\text{uLX[0]} = (\text{ mvpLX[0]} + \text{mvdLX[0]} + 2^{16}\text{) \% } 2^{16} \quad \text{(I-110)}$$

   $$\text{mvLX[0]} = (\text{ uLX[0]} >= 2^{15}\text{) ? (uLX[0]} - 2^{16}\text{) : uLX[0]} \quad \text{(I-111)}$$

   $$\text{uLX[1]} = (\text{ mvpLX[1]} + \text{mvdLX[1]} + 2^{16}\text{) \% } 2^{16} \quad \text{(I-112)}$$

   $$\text{mvLX[1]} = (\text{ uLX[1]} >= 2^{15}\text{) ? (uLX[1]} - 2^{16}\text{) : uLX[1]} \quad \text{(I-113)}$$

NOTE – The resulting values of mvLX[0] and mvLX[1] as specified above will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

When ChromaArrayType is not equal to 0 and predFlagLX, with X being 0 or 1, is equal to 1, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with mvLX as input, and the output being mvCLX.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xPb..(xPb + nPbW − 1), y = yPb..(yPb + nPbH − 1):

$$\text{IvMcFlag[x][y]} = \text{ivMcFlag} \quad \text{(I-114)}$$
$$\text{VspMcFlag[x][y]} = \text{vspMcFlag} \quad \text{(I-115)}$$

I.8.5.3.2.2 Derivation process for an initial merge candidate list

The specifications in clause 8.5.3.2.2 apply with the following modifications:

– Steps 9 and 10 are removed.

– "When slice_type is equal to B, the derivation process for combined bi-predictive merging candidates" is replaced with "When slice_type is equal to B and numCurrMergeCand is less than 5, the derivation process for combined bi-predictive merging candidates".

– "derivation process for merging candidates from neighbouring prediction unit partitions in clause 8.5.3.2.3" is replaced with "derivation process for merging candidates from neighbouring prediction unit partitions in clause I.8.5.3.2.3".

– "temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked" is replaced with "temporal luma motion vector prediction in clause I.8.5.3.2.5 is invoked".

– The outputs of the process are replaced with:

  – the modified luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

  – the variables nPbW and nPbH specifying the modified width and the height of the luma prediction block,

  – the modified variable partIdx specifying the modified index of the current prediction unit within the current coding unit,

  – the original luma location (xOrigP, yOrigP) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

  – the variables nOrigPbW and nOrigPbH specifying the original width and the height of the luma prediction block,

  – the merging candidate list, mergeCandList,

  – the luma motion vectors mvL0N and mvL1N, with N being replaced by all elements of mergeCandList,

  – the reference indices refIdxL0N and refIdxL1N, with N being replaced by all elements of mergeCandList,

  – the prediction list utilization flags predFlagL0N and predFlagL1N, with N being replaced by all elements of mergeCandList.

I.8.5.3.2.3 Derivation process for spatial merging candidates

The specifications in clause 8.5.3.2.3 apply with the following modifications and additions:

– The function differentMotionLoc(xN, yN, xM, yM) is specified as follows:

  – If one or more of the following conditions are true, for X in the range of 0 to 1, inclusive, differentMotionLoc(xN, yN, xM, yM) is set equal to 1:

    – PredFlagLX[xN][yN] is not equal to PredFlagLX[xM][yM].

    – MvLX[xN][yN] is not equal to MvLX[xM][yM].

    – RefIdxLX[xN][yN] is not equal to RefIdxLX[xM][yM].

  – Otherwise, differentMotionLoc(xN, yN, xM, yM) is set equal to 0.

– "the prediction units covering the luma locations ($xNbA_1$, $yNbA_1$) and ($xNbB_1$, $yNbB_1$) have the same motion vectors and the same reference indices" is replaced with "differentMotionLoc($xNbA_1$, $yNbA_1$, $xNbB_1$, $yNbB_1$) is equal to 0".

– "the prediction units covering the luma locations ($xNbB_1$, $yNbB_1$) and ($xNbB_0$, $yNbB_0$) have the same motion vectors and the same reference indices" is replaced with "differentMotionLoc($xNbB_1$, $yNbB_1$, $xNbB_0$, $yNbB_0$) is equal to 0".

– "the prediction units covering the luma locations ($xNbA_1$, $yNbA_1$) and ($xNbA_0$, $yNbA_0$) have the same motion vectors and the same reference indices" is replaced with "differentMotionLoc($xNbA_1$, $yNbA_1$, $xNbA_0$, $yNbA_0$) is equal to 0".

– "prediction units covering the luma locations ($xNbA_1$, $yNbA_1$) and ($xNbB_2$, $yNbB_2$) have the same motion vectors and the same reference indices" is replaced with "differentMotionLoc($xNbA_1$, $yNbA_1$, $xNbB_2$, $yNbB_2$) is equal to 0".

– "the prediction units covering the luma locations ($xNbB_1$, $yNbB_1$) and ($xNbB_2$, $yNbB_2$) have the same motion vectors and the same reference indices" is replaced with "differentMotionLoc($xNbB_1$, $yNbB_1$, $xNbB_2$, $yNbB_2$) is equal to 0".

I.8.5.3.2.4 Derivation process for luma motion vector prediction

The specifications in clause 8.5.3.2.6 apply with the following modifications:

– "temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked" is replaced by "temporal luma motion vector prediction in clause I.8.5.3.2.5 is invoked".

I.8.5.3.2.5 Derivation process for temporal luma motion vector prediction

Inputs to this process are:

– a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

– the motion vector prediction mvLXCol,

– the availability flag availableFlagLXCol.

The variable currPb specifies the current luma prediction block at luma location (xPb, yPb).

The variables mvLXCol and availableFlagLXCol are derived as follows:

– If slice_temporal_mvp_enabled_flag is equal to 0, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

– Otherwise, the following ordered steps apply:

1. The bottom right collocated motion vector is derived as follows:

$$xColBr = xPb + nPbW \quad \text{(I-116)}$$

$$yColBr = yPb + nPbH \quad \text{(I-117)}$$

– If yPb >> CtbLog2SizeY is equal to yColBr >> CtbLog2SizeY, yColBr is less than pic_height_in_luma_samples and xColBr is less than pic_width_in_luma_samples, the following applies:

  – The luma location (xColPb, yColPb) is set equal to ((xColBr >> 4) << 4, (yColBr >> 4) << 4).

  – The variable colPb specifies the luma prediction block covering the modified location given by (xColPb, yColPb) inside the collocated picture specified by ColPic.

  – Depending on merge_flag[xPb][yPb], the following applies:

    – If merge_flag[xPb][yPb] is equal to 0, the derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

    – Otherwise (merge_flag[xPb][yPb] is equal to 1), the derivation process for collocated motion vectors for merge mode as specified in clause I.8.5.3.2.6 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

– Otherwise, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

2. When availableFlagLXCol is equal to 0, the central collocated motion vector is derived as follows:

$$xColCtr = xPb + (nPbW >> 1) \quad \text{(I-118)}$$

$$yColCtr = yPb + (nPbH >> 1) \quad \text{(I-119)}$$

– The luma location (xColPb, yColPb) is set equal to ((xColCtr >> 4) << 4, (yColCtr >> 4) << 4).

– The variable colPb specifies the luma prediction block covering the modified location given by (xColPb, yColPb) inside the collocated picture specified by ColPic.

– Depending on merge_flag[xPb][yPb], the following applies:

  – If merge_flag[xPb][yPb] is equal to 0, the derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

  – Otherwise (merge_flag[xPb][yPb] is equal to 1), the derivation process for collocated motion vectors for merge mode as specified in clause I.8.5.3.2.6 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

I.8.5.3.2.6 Derivation process for collocated motion vectors for merge mode

Inputs to this process are:

- a variable currPb specifying the current prediction block,
- a variable colPb specifying the collocated prediction block inside the collocated picture specified by ColPic,
- a luma location (xColPb, yColPb) specifying a sample inside the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic,
- a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

- the motion vector prediction mvLXCol,
- the availability flag availableFlagLXCol.

The arrays predFlagL0Col[x][y], mvL0Col[x][y], and refIdxL0Col[x][y] are set equal to PredFlagL0[x][y], MvL0[x][y], and RefIdxL0[x][y], respectively, of the collocated picture specified by ColPic, and the arrays predFlagL1Col[x][y], mvL1Col[x][y], and refIdxL1Col[x][y] are set equal to PredFlagL1[x][y], MvL1[x][y], and RefIdxL1[x][y], respectively, of the collocated picture specified by ColPic.

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If colPb is coded in an intra prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the following applies:
  - The motion vector mvCol, the reference index refIdxCol, and the reference list identifier listCol are derived as follows:
    - If predFlagL0Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol, and listCol are set equal to mvL1Col[xColPb][yColPb], refIdxL1Col[xColPb][yColPb], and L1, respectively.
    - Otherwise, if predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol, and listCol are set equal to mvL0Col[xColPb][yColPb], refIdxL0Col[xColPb][yColPb], and L0, respectively.
    - Otherwise (predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 1), the following assignments are made:
      - If NoBackwardPredFlag is equal to 1, mvCol, refIdxCol, and listCol are set equal to mvLXCol[xColPb][yColPb], refIdxLXCol[xColPb][yColPb], and LX, respectively.
      - Otherwise, mvCol, refIdxCol, and listCol are set equal to mvLNCol[xColPb][yColPb], refIdxLNCol[xColPb][yColPb], and LN, respectively, with N being the value of collocated_from_l0_flag.
  - The motion vector mvLXCol and availableFlagLXCol are derived as follows:
    - The variables curLtFlag and colLtFlag are derived as follows:

      curLtFlag = LongTermRefPic(CurrPic, currPb, refIdxLX, LX) (I-120)

      colLtFlag = LongTermRefPic(ColPic, colPb, refIdxCol, listCol) (I-121)
    - When curLtFlag is not equal to colLtFlag and AltRefIdxLX is not equal to −1, the variables AltRefFlagLX, refIdxLX, and curLtFlag are modified as follows:

      AltRefFlagLX = 1 (I-122)

      refIdxLX = AltRefIdxLX (I-123)

      curLtFlag = LongTermRefPic(CurrPic, currPb, refIdxLX, LX) (I-124)
    - The motion vector mvLXCol is modified as follows:
      - If curLtFlag is not equal to colLtFlag, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
      - Otherwise, the variable availableFlagLXCol is set equal to 1, refPicListCol[refIdxCol] is set to be the picture with reference index refIdxCol in the reference picture list listCol of the slice containing

prediction block colPb in the collocated picture specified by ColPic, and the following applies:

– The variables colDiff and currDiff are set equal to 0 and modified as follows:

  – If curLtFlag is equal to 0, the following applies:

$$\text{colDiff} = \text{DiffPicOrderCnt}(\ \text{ColPic}, \text{refPicListCol}[\ \text{refIdxCol}\ ]\ ) \quad \text{(I-125)}$$

$$\text{currDiff} = \text{DiffPicOrderCnt}(\ \text{CurrPic}, \text{RefPicListX}[\ \text{refIdxLX}\ ]\ ) \quad \text{(I-126)}$$

  – Otherwise (curLtFlag is equal to 1), when IvMvScalEnabledFlag is equal to 1, the following applies:

$$\text{colDiff} = \text{ViewIdVal}(\ \text{ColPic}\ ) - \text{ViewIdVal}(\ \text{refPicListCol}[\ \text{refIdxCol}\ ]\ ) \quad \text{(I-127)}$$

$$\text{currDiff} = \text{ViewIdVal}(\ \text{CurrPic}\ ) - \text{ViewIdVal}(\ \text{RefPicListX}[\ \text{refIdxLX}\ ]\ ) \quad \text{(I-128)}$$

– Depending on the values of colDiff and currDiff, the following applies:

  – If colDiff is equal to currDiff, or colDiff is equal to 0, or currDiff is equal to 0, mvLXCol is derived as follows:

$$\text{mvLXCol} = \text{mvCol} \quad \text{(I-129)}$$

  – Otherwise, mvLXCol is derived as a scaled version of the motion vector mvCol as follows:

$$\text{tx} = (\ 16384 + (\ \text{Abs}(\ \text{td}\ ) >> 1\ )\ ) / \text{td} \quad \text{(I-130)}$$

$$\text{distScaleFactor} = \text{Clip3}(\ -4096, 4095, (\ \text{tb} * \text{tx} + 32\ ) >> 6\ ) \quad \text{(I-131)}$$

$$\text{mvLXCol} = \text{Clip3}(\ -32768, 32767, \text{Sign}(\ \text{distScaleFactor} * \text{mvCol}\ ) * (\ (\ \text{Abs}(\ \text{distScaleFactor} * \text{mvCol}\ ) + 127\ ) >> 8\ )\ ) \quad \text{(I-132)}$$

  where td and tb are derived as follows:

$$\text{td} = \text{Clip3}(\ -128, 127, \text{colDiff}\ ) \quad \text{(I-133)}$$

$$\text{tb} = \text{Clip3}(\ -128, 127, \text{currDiff}\ ) \quad \text{(I-134)}$$

I.8.5.3.2.7 Derivation process for luma motion vectors for merge mode

This process is only invoked when merge_flag[xPb][yPb] is equal to 1, where (xPb, yPb) specifies the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– the luma motion vectors mvL0 and mvL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1,

– the flag ivMcFlag specifying whether the current PU is coded using an inter-view predicted merge candidate,

– the flag vspMcFlag specifying whether the current PU is coded using the view synthesis prediction merge candidate,

– the flag subPbMotionFlag specifying whether the current PU is coded using sub-block partitions.

The function differentMotion(N, M) is specified as follows:

– If one or more of the following conditions are true, for X in the range of 0 to 1, inclusive, differentMotion(N, M) is set equal to 1:

  - predFlagLXN is not equal to predFlagLXM.
  - mvLXN is not equal to mvLXM.
  - refIdxLXN is not equal to refIdxLXM.
- Otherwise, differentMotion(N, M) is set equal to 0.

The variables AltRefFlagL0 and AltRefFlagL1 are set equal to 0.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, and the prediction utilization flags predFlagL0 and predFlagL1 are derived by the following ordered steps:

1. The derivation process for an initial merge candidate list as specified in clause I.8.5.3.2.2 is invoked with the luma location (xCb, yCb), the luma location (xPb, yPb), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the outputs being a modified luma location (xPb, yPb), the modified variables nPbW and nPbH, the modified variable partIdx, the luma location (xOrigP, yOrigP), the variables nOrigPbW and nOrigPbH, the merging candidate list initMergeCandList, the luma motion vectors mvL0N and mvL1N, the reference indices refIdxL0N and refIdxL1N, and the prediction list utilization flags predFlagL0N and predFlagL1N, with N being replaced by all elements of initMergeCandList.

2. For N being replaced by A_1, B_1, B_0, A_0 and B_2, the following applies:
   - If N is an element in initMergeCandList, availableFlagN is set equal to 1.
   - Otherwise (N is not an element in initMergeCandList), availableFlagN is set equal to 0.

3. Depending on the values of IvDiMcEnabledFlag, DispAvailFlag, and IlluCompFlag[xCb][yCb], the following applies:
   - If IvDiMcEnabledFlag is equal to 0, DispAvailFlag is equal to 0, or IlluCompFlag[xCb][yCb] is equal to 1, the flags availableFlagIV and availableFlagIVShift are set equal to 0.
   - Otherwise (IvDiMcEnabledFlag is equal to 1, DispAvailFlag is equal to 1, and IlluCompFlag[xCb][yCb] is equal to 0), the derivation process for inter-view predicted merging candidates as specified in clause I.8.5.3.2.8 is invoked with the luma location (xPb, yPb), and the variables nPbW and nPbH as inputs, and the availability flags availableFlagIV and availableFlagIVShift, the prediction list utilization flags predFlagLXIV and predFlagLXIVShift, the reference indices refIdxLXIV and refIdxLXIVShift, and the motion vectors mvLXIV and mvLXIVShift (with X in the range of 0 to 1, inclusive) as outputs.

4. Depending on the value of TexMcEnabledFlag, the texture merging candidate is derived as follows:
   - If TexMcEnabledFlag is equal to 0, the variable availableFlagT is set equal to 0.
   - Otherwise (TexMcEnabledFlag is equal to 1), the following applies:
     - The derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate as specified in clause I.8.5.3.2.9 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the merging candidate indicator N equal to T as inputs, and the prediction utilization flag predFlagLXT, the reference index refIdxLXT, and the motion vector mvLXT (with X in the range of 0 to 1, inclusive) as outputs.
     - The flag availableFlagT is set equal to (predFlagL0T | | predFlagL1T).

5. Depending on the values of IvDiMcEnabledFlag, DispAvailFlag, and DepthFlag, the following applies:
   - If IvDiMcEnabledFlag is equal to 0, DispAvailFlag is equal to 0, or DepthFlag is equal to 1, the flags availableFlagDI and availableFlagDIShift are set equal to 0.
   - Otherwise (IvDiMcEnabledFlag is equal to 1, DispAvailFlag is equal to 1, and DepthFlag is equal to 0), the derivation process for disparity information merging candidates as specified in clause I.8.5.3.2.12 is invoked with the luma location (xPb, yPb), and the variables nPbW and nPbH as inputs, and the availability flags availableFlagDI and availableFlagDIShift, the prediction list utilization flags predFlagLXDI and predFlagLXDIShift, the reference indices refIdxLXDI and refIdxLXDIShift, and the motion vectors mvLXDI and mvLXDIShift (with X in the range of 0 to 1, inclusive) as outputs.

6. Depending on the values of VspMcEnabledFlag, DispAvailFlag, IlluCompFlag[xCb][yCb], iv_res_pred_weight_idx[xCb][yCb], and DbbpFlag[xCb][yCb], the following applies:
   - If one or more of the following conditions are true, the flag availableFlagVSP is set equal to 0:
     - VspMcEnabledFlag is equal to 0.

– DispAvailFlag is equal to 0.

– IlluCompFlag[xCb][yCb] is equal to 1.

– iv_res_pred_weight_idx[xCb][yCb] is not equal to 0.

– DbbpFlag[xCb][yCb] is equal to 1.

– Otherwise, the derivation process for a view synthesis prediction merging candidate as specified in clause I.8.5.3.2.13 is invoked with the luma locations (xPb, yPb) and the variables nPbW and nPbH as inputs, and the availability flag availableFlagVSP, the prediction list utilization flag predFlagLXVSP, the reference index refIdxLXVSP, and the motion vector mvLXVSP (with X in the range of 0 to 1, inclusive) as outputs.

7. The merging candidate list extMergeCandList is constructed as follows:

```
i = 0
if( availableFlagT )
    extMergeCandList[ i++ ] = T
if( availableFlagIV  &&  ( !availableFlagT  ||  differentMotion( T, IV ) ) )
    extMergeCandList[ i++ ] = IV
N = DepthFlag ? T : IV
if( availableFlagA1  &&  ( !availableFlagN  ||  differentMotion( N, A1 ) ) )
    extMergeCandList[ i++ ] = A1
if( availableFlagB1  &&  ( !availableFlagN  ||  differentMotion( N, B1 ) ) )
    extMergeCandList[ i++ ] = B1
if( availableFlagVSP  &&  ( !availableFlagA1  ||  !VspMcFlag[ xPb − 1 ][ yPb + nPbH − 1 ] )  &&
    i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = VSP
if( availableFlagB0 )
    extMergeCandList[ i++ ] = B0                                              (I-135)
if( availableFlagDI  &&  ( !availableFlagA1  ||  differentMotion( A1, DI ) )  &&
    ( !availableFlagB1  ||  differentMotion( B1, DI ) )  &&  ( i < MaxNumMergeCand ) )
    extMergeCandList[ i++ ] = DI
if( availableFlagA0  &&  i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = A0
if( availableFlagB2  &&  i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = B2
if( availableFlagIVShift  &&  i < MaxNumMergeCand  &&
                    ( !availableFlagIV  ||  differentMotion( IV, IVShift ) ) )
    extMergeCandList[ i++ ] = IVShift
if( availableFlagDIShift  &&  !availableFlagIVShift  &&  i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = DIShift
j = 0
while( i < MaxNumMergeCand ) {                                                (I-136)
    N = initMergeCandList[ j++ ]
    if( N  !=  A1  &&  N  !=  B1  &&  N  !=  B0  &&  N  !=  A0  &&  N  !=  B2 )
        extMergeCandList[ i++ ] = N
}
```

8. The variable N is derived as follows:

– If (nOrigPbW + nOrigPbH) is equal to 12, the following applies:

N = initMergeCandList[merge_idx[xOrigP][yOrigP]] (I-137)

– Otherwise, ((nOrigPbW + nOrigPbH) is not equal to 12), the following applies:

N = extMergeCandList[merge_idx[xOrigP][yOrigP]] (I-138)

9. When N is equal to Col, the following applies for X in the range of 0 to 1, inclusive:

– When AltRefFlagLX is equal to 1, refIdxLXCol is set equal to AltRefIdxLX.

10. When availableFlagVSP is equal to 1, N is equal to A_1, and VspMcFlag[xPb − 1][yPb + nPbH − 1] is equal to 1, N is set equal to VSP.

11. The variable subPbMotionFlag is derived as follows:

subPbMotionFlag = (!DepthFlag && !DbbpFlag[xCb][yCb] && N = = IV)
|| N = = VSP || N = = T (I-139)

12. Depending on the value of subPbMotionFlag, the following applies:

   – If subPbMotionFlag is equal to 0, the following applies, for X in the range of 0 to 1, inclusive:

     mvLX = mvLXN (I-140)

     refIdxLX = refIdxLXN (I-141)

     predFlagLX = predFlagLXN (I-142)

   – Otherwise (subPbMotionFlag is equal to 1), SubPbArrayPartSize is set equal to SubPbArrayPartSizeN and the following applies for X in the range of 0 to 1, inclusive:

     mvLX = (0, 0) (I-143)

     SubPbArrayMvLX = SubPbArrayMvLXN (I-144)

     SubPbArrayMvCLX = SubPbArrayMvCLXN (I-145)

     refIdxLX = −1 (I-146)

     SubPbArrayRefIdxLX = SubPbArrayRefIdxLXN (I-147)

     predFlagLX = 0 (I-148)

     SubPbArrayPredFlagLX = SubPbArrayPredFlagLXN (I-149)

   NOTE – When subPbMotionFlag is equal to 1, luma motion vectors, chroma motion vectors, reference indices, and prediction utilization flags are given in sub-block partition granularity in the arrays SubPbArrayPredFlagLX, SubPbArrayMvLX, SubPbArrayMvCLX, SubPbArrayRefIdxLX (with X in the range of 0 to 1, inclusive). Moreover, the width and height of the sub-block partitions is stored in the array SubPbArrayPartSize.

13. When all of the following conditions are true, refIdxL1 is set equal to −1 and predFlagL1 is set equal to 0:

   – predFlagL0 and predFlagL1 are equal to 1.

   – (nOrigPbW + nOrigPbH) is equal to 12 or DbbpFlag[xCb][yCb] is equal to 1.

14. The flag ivMcFlag is set equal to (N = = IV || N = = IVShift).

15. The flag vspMcFlag is set equal to (N = = VSP).

I.8.5.3.2.8 Derivation process for inter-view predicted merging candidates

Inputs to this process are:

– a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the current luma prediction block.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

– the availability flags availableFlagIV and availableFlagIVShift specifying whether the inter-view predicted and shifted inter-view predicted merging candidates are available,

– the prediction list utilization flags predFlagLXIV and predFlagLXIVShift,

– the reference indices refIdxLXIV and refIdxLXIVShift,

– the motion vectors mvLXIV and mvLXIVShift.

The availability flags availableFlagIV and availableFlagIVShift are set equal to 0, and for X in the range of 0 to 1, inclusive, the following applies:

– The variables predFlagLXIV and predFlagLXIVShift are set equal to 0, the variables refIdxLXIV and refIdxLXIVShift are set equal to −1, and the motion vectors mvLXIV and mvLXIVShift are set equal to (0, 0).

The inter-view predicted merging candidate is derived by the following ordered steps:

1. Depending on the values of DbbpFlag[xPb][yPb] and DepthFlag, the following applies:

   – If DbbpFlag[xPb][yPb] is equal to 0 and DepthFlag is equal to 0, the derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate as specified in clause I.8.5.3.2.9 is

invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the merging candidate indicator N equal to IV as inputs, and the outputs are the flag predFlagLXIV, the reference index refIdxLXIV, and the motion vector mvLXIV (with X in the range of 0 to 1, inclusive).

– Otherwise (DbbpFlag[xPb][yPb] is not equal to 0 or DepthFlag is equal to 1), the derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the disparity vector offset (xOff, yOff) equal to (0, 0) as inputs, and the outputs are the flag predFlagLXIV, the reference index refIdxLXIV, and the motion vector mvLXIV (with X in the range of 0 to 1, inclusive).

2. The availability flag availableFlagIV is set equal to (predFlagL0IV || predFlagL1IV).

When DepthFlag is equal to 0, the shifted inter-view predicted merging candidate is derived by the following ordered steps:

1. The derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the disparity vector offset (xOff, yOff) equal to (nPbW * 2, nPbH * 2) as inputs, and the outputs are the flag predFlagLXIVShift, the reference index refIdxLXIVShift, and the motion vector mvLXIVShift (with X in the range of 0 to 1, inclusive).

2. The availability flag availableFlagIVShift is set equal to (predFlagL0IVShift || predFlagL1IVShift).

I.8.5.3.2.9 Derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate

Inputs to this process are:

– a luma location (xPb, yPb) of the top-left sample of the current prediction luma block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the current prediction block,

– a merging candidate indicator N.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

– the prediction utilization flag predFlagLXN,

– the reference index refIdxLXN,

– the motion vector mvLXN.

For X in the range of 0 to 1, inclusive, the variable predFlagLXN is set equal to 0, the variable refIdxLXN is set equal to −1, the motion vector mvLXN is set equal to (0, 0).

The variables nSbW and nSbH specifying the width and height of the sub-block partitions and the luma location (xSbDef, ySbDef) of the default sub-block partition are derived as follows:

$$\text{subPbTmcSize} = 1 \ll (\text{log2\_texmc\_sub\_pb\_size\_minus3}[\text{DepthFlag}] + 3) \quad \text{(I-150)}$$

$$\text{subPbIvMcSize} = 1 \ll (\text{log2\_ivmc\_sub\_pb\_size\_minus3}[\text{DepthFlag}] + 3) \quad \text{(I-151)}$$

$$\text{minSize} = (\text{N} == \text{T}) \mathbin{?} \text{subPbTmcSize} : \text{subPbIvMcSize} \quad \text{(I-152)}$$

$$\text{nSbW} = (\text{nPbW} \,\%\, \text{minSize} \mathrel{!=} 0 \;||\; \text{nPbH} \,\%\, \text{minSize} \mathrel{!=} 0) \mathbin{?} \text{nPbW} : \text{minSize} \quad \text{(I-153)}$$

$$\text{nSbH} = (\text{nPbW} \,\%\, \text{minSize} \mathrel{!=} 0 \;||\; \text{nPbH} \,\%\, \text{minSize} \mathrel{!=} 0) \mathbin{?} \text{nPbH} : \text{minSize} \quad \text{(I-154)}$$

$$(\text{xSbDef}, \text{ySbDef}) = (\text{xPb} + ((\text{nPbW} / \text{nSbW}) / 2) * \text{nSbW}, \text{yPb} + ((\text{nPbH} / \text{nSbH}) / 2) * \text{nSbH}) \quad \text{(I-155)}$$

Depending on the value of N, the variables predFlagLXN, refIdxLXN, and mvLXN are derived as follows:

– If N is equal to IV, the derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xSbDef, ySbDef), the variables nSbW and nSbH, and the disparity vector offset (xOff, yOff) equal to (0, 0) as inputs, and the outputs are the flag predFlagLXN, the reference index refIdxLXN, and the motion vector mvLXN (with X in the range of 0 to 1, inclusive).

– Otherwise (N is equal to T), the derivation process for motion vectors for the texture merge candidate as specified in clause I.8.5.3.2.11 is invoked with the luma location (xSbDef, ySbDef), and the variables nSbW and nSbH as inputs, and the outputs are the flags predFlagLXN, the reference index refIdxLXN, and the motion vector mvLXN (with X in the range of 0 to 1, inclusive).

When predFlagL0N or predFlagL1N is equal to 1, the following applies:

– For yBlk in the range of 0 to (nPbH / nSbH − 1), inclusive, the following applies:
  – For xBlk in the range of 0 to (nPbW / nSbW − 1), inclusive, the following applies:
    – The luma location (xSb, ySb) of the current sub-block partition is derived as follows:

$$(xSb, ySb) = (xPb + xBlk * nSbW, yPb + yBlk * nSbH) \quad \text{(I-156)}$$

    – Depending on the value of N, the following applies:
      – If N is equal to IV, the derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xSb, ySb), the variables nSbW and nSbH, and the disparity vector offset (xOff, yOff) equal to (0, 0) as inputs, and the outputs are the flag spPredFlagLX[xBlk][yBlk], the reference index spRefIdxLX[xBlk][yBlk], and the motion vector spMvLX[xBlk][yBlk] (with X in the range of 0 to 1, inclusive).
      – Otherwise (N is equal to T), the derivation process for motion vectors for the texture merge candidate as specified in clause I.8.5.3.2.11 is invoked with the luma location (xSb, ySb), and the variables nSbW and nSbH as inputs, and the outputs are the flags spPredFlagLX[xBlk][yBlk], the reference index spRefIdxLX[xBlk][yBlk], and the motion vector spMvLX[xBlk][yBlk] (with X in the range of 0 to 1, inclusive).
    – When spPredFlagL0[xBlk][yBlk] and spPredFlagL1[xBlk][yBlk] are both equal to 0, the following applies for X in the range of 0 to 1, inclusive:

$$spPredFlagLX[xBlk][yBlk] = predFlagLXN \quad \text{(I-157)}$$

$$spRefIdxLX[xBlk][yBlk] = refIdxLXN \quad \text{(I-158)}$$

$$spMvLX[xBlk][yBlk] = mvLXN \quad \text{(I-159)}$$

– For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = 0..nPbW − 1 and y = 0..nPbH − 1:
  – The variable SubPbArrayPartSizeN is derived as follows:

$$SubPbArrayPartSizeN[xPb + x][yPb + y] = (nSbW, nSbH) \quad \text{(I-160)}$$

  – For X in the range of 0 to 1, inclusive, the following applies:
    – The variables SubPbArrayPredFlagLX, SubPbArrayMvLX, and SubPbArrayRefIdxLX are derived as follows:

$$SubPbArrayPredFlagLXN[xPb + x][yPb + y] = spPredFlagLX[x / nSbW][y / nSbH] \quad \text{(I-161)}$$

$$SubPbArrayRefIdxLXN[xPb + x][yPb + y] = spRefIdxLX[x / nSbW][y / nSbH] \quad \text{(I-162)}$$

$$SubPbArrayMvLXN[xPb + x][yPb + y] = spMvLX[x / nSbW][y / nSbH] \quad \text{(I-163)}$$

    – The derivation process for chroma motion vectors as specified in clause 8.5.3.2.10 is invoked with SubPbArrayMvLXN[xPb + x][yPb + y] as input, and the output is SubPbArrayMvCLXN[xPb + x][yPb + y].

I.8.5.3.2.10 Derivation process for motion vectors for an inter-view predicted merging candidate

Inputs to this process are:

– a luma location (xBlk, yBlk) of the top-left sample of the current luma prediction block or sub-block partition relative to the top-left luma sample of the current picture,
– two variables nBlkW and nBlkH specifying the width and the height of the current luma prediction block or sub-block partition,
– a disparity vector offset (xOff, yOff).

Outputs of this process are (with X in the range of 0 to 1, inclusive):

– the prediction utilization flag predFlagLXInterView,
– the reference index refIdxLXInterView,
– the motion vector mvLXInterView.

For X in the range of 0 to 1, inclusive, the flag predFlagLXInterView is set equal to 0, the variable refIdxLXInterView is set equal to −1, and the motion vector mvLXInterView is set equal to (0, 0).

The luma location (xRef, yRef) is derived as follows:

dispVec[0] = DispRefVec[xBlk][yBlk][0] + xOff (I-164)

dispVec[1] = DispRefVec[xBlk][yBlk][1] + yOff (I-165)

xRefFull = xBlk + (nBlkW >> 1) + ((dispVec[0] + 2) >> 2) (I-166)

yRefFull = yBlk + (nBlkH >> 1) + ((dispVec[1] + 2) >> 2) (I-167)

xRef = Clip3(0, pic_width_in_luma_samples − 1, (xRefFull >> 3) << 3) (I-168)

yRef = Clip3(0, pic_height_in_luma_samples − 1, (yRefFull >> 3) << 3) (I-169)

Let ivRefPic the picture with nuh_layer_id equal to ViewCompLayerId[RefViewIdx[xBlk][yBlk]][DepthFlag] in the current access unit and let ivRefPb be the luma prediction block covering the luma location (xRef, yRef) in the picture ivRefPic.

The luma location (xIvRefPb, yIvRefPb) is set equal to the location of the top-left sample of ivRefPb relative to the top-left luma sample of the picture ivRefPic.

When ivRefPb is not coded in an intra prediction mode, the following applies for X in the range of 0 to (slice_type = = B) ? 1 : 0, inclusive:

– For k in the range of 0 to 1, inclusive, the following applies:
  – Y is set equal to (k = = 0) ? X : (1 − X).
  – The variables predFlagLYIvRef[x][y], mvLYIvRef[x][y], and refIdxLYIvRef[x][y] are set equal to PredFlagLY[x][y], MvLY[x][y], and RefIdxLY[x][y], respectively, of picture ivRefPic.
  – The variable refPicListYIvRef is set equal to RefPicListY of the slice containing ivRefPb in picture ivRefPic.
  – When predFlagLYIvRef[xIvRefPb][yIvRefPb] is equal to 1, the following applies for i in the range of 0 to num_ref_idx_lX_active_minus1, inclusive:
    – When PicOrderCnt(refPicListYIvRef[refIdxLYIvRef[xIvRefPb][yIvRefPb]]) is equal to PicOrderCnt(RefPicListX[i]) and predFlagLXInterView is equal to 0, the following applies:

predFlagLXInterView = 1 (I-170)

refIdxLXInterView = i (I-171)

mvLXInterView = mvLYIvRef[xIvRefPb][yIvRefPb] (I-172)

I.8.5.3.2.11 Derivation process for motion vectors for the texture merge candidate

Inputs to this process are:

– a luma location (xSb, ySb) of the top-left sample of the current luma sub-block partition relative to the top-left luma sample of the current picture,
– two variables nSbW and nSbH specifying the width and the height of the current sub-block partition,

Outputs of this process are (with X in the range of 0 to 1, inclusive):

– the prediction utilization flag predFlagLXT,
– the reference index refIdxLXT,
– the motion vector mvLXT.

For X in the range of 0 to 1, inclusive, the variable predFlagLXT is set equal to 0, the variable refIdxLXT is set equal to −1, and the motion vector mvLX0T is set equal to (0, 0).

The texture luma location (xRef, yRef) is derived as follows:

xRefFull = xSb + (nSbW >> 1) (I-173)

yRefFull = ySb + (nSbH >> 1) (I-174)

xRef = (xRefFull >> 3) << 3 (I-175)

yRef = (yRefFull >> 3) << 3 (I-176)

Let textPb be the prediction block covering the luma sample location (xRef, yRef) in the picture TexturePic.

For X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive, the following applies:

– The arrays textPredFlagLX[x][y], textRefIdxLX[x][y], and textMvLX[x][y] are set equal to PredFlagLX[x][y], RefIdxLX[x][y], and MvLX[x][y], respectively, of the picture TexturePic.

– The list textRefPicListX is set equal to RefPicListX of the slice containing textPb in the picture TexturePic.

– When textPredFlagLX[xRef][yRef] is equal to 1, the following applies for i in the range of 0 to num_ref_idx_lX_active_minus1, inclusive:

  – When PicOrderCnt(RefPicListX[i]) is equal to PicOrderCnt(textRefPicListX[textRefIdxLX]), ViewIdx(RefPicListX[i]) is equal to ViewIdx(textRefPicListX[textRefIdxLX]), and predFlagLXT is equal to 0, the following applies:

$$\text{predFlagLXT} = 1 \tag{I-177}$$

$$\text{refIdxLXT} = \text{i} \tag{I-178}$$

$$\text{mvLXT[0]} = (\text{ textMvLX[xRef][yRef][0]} + 2\ ) >> 2 \tag{I-179}$$

$$\text{mvLXT[1]} = (\text{ textMvLX[xRef][yRef][1]} + 2\ ) >> 2 \tag{I-180}$$

I.8.5.3.2.12 Derivation process for disparity information merging candidates

Inputs to this process are:

– a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the current prediction block.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

– the flags availableFlagDI and availableFlagDIShift, specifying whether the disparity information merging candidate and the shifted disparity information candidate are available,

– the prediction list utilization flags predFlagLXDI and predFlagLXDIShift,

– the reference indices refIdxLXDI and refIdxLXDIShift,

– the motion vectors mvLXDI and mvLXDIShift.

The variables availableFlagDI and availableFlagDIShift are set equal to 0, and for X in the range of 0 to 1, inclusive, the following applies:

– The variables predFlagLXDI and predFlagLXDIShift are set equal to 0, the variables refIdxLXDI and refIdxLXDIShift are set equal to −1, and the motion vectors mvLXDI and mvLXDIShift are set equal to (0, 0).

The disparity information merging candidate is derived as follows:

– For X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive, the following applies:

  – For i in the range of 0 to num_ref_idx_lX_active_minus1, inclusive, the following applies:

    – When PicOrderCnt(RefPicListX[i]) is equal to the PicOrderCntVal, ViewIdx(RefPicListX[i]) is equal to RefViewIdx[xPb][yPb], and predFlagLXDI is equal to 0, the following applies:

$$\text{availableFlagDI} = 1 \tag{I-181}$$

$$\text{predFlagLXDI} = 1 \tag{I-182}$$

$$\text{refIdxLXDI} = \text{i} \tag{I-183}$$

$$\text{mvLXDI} = (\text{ DispRefVec[xPb][yPb][0]}, 0\ ) \tag{I-184}$$

When availableFlagDI is equal to 1, the shifted disparity information merging candidate is derived as follows:

– The variable availableFlagDIShift is set equal to 1.

– The following applies for X in the range of 0 to 1, inclusive:

$$\text{predFlagLXDIShift} = \text{predFlagLXDI} \tag{I-185}$$

$$\text{refIdxLXDIShift} = \text{refIdxLXDI} \tag{I-186}$$

$$\text{mvLXDIShift} = \text{predFlagLXDI} \ ?\ (\text{ mvLXDI[0]} + 4, \text{mvLXDI[1]}\ ) : (\ 0, 0\ ) \tag{I-187}$$

I.8.5.3.2.13 Derivation process for a view synthesis prediction merging candidate

Inputs to this process are:

– a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the current prediction block.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

– the availability flag availableFlagVSP specifying whether the view synthesis prediction merging candidate is available,

– the prediction list utilization flag predFlagLXVSP,

– the reference index refIdxLXVSP,

– the motion vector mvLXVSP.

The variable availableFlagVSP is set equal to 0 and for X in the range of 0 to 1, inclusive, the following applies:

– The variable predFlagLXVSP is set equal to 0, the variable refIdxLXVSP is set equal to −1, and the motion vector mvLXVSP is set equal to (0, 0).

For X in the range of 0 to (slice_type = = B ? 1 : 0), inclusive, the following applies:

– For i in the range of 0 to num_ref_idx_lX_active_minus1, inclusive, the following applies:

  – When availableFlagVSP is equal to 0 and ViewIdx(RefPicListX[i]) is equal to RefViewIdx[xPb][yPb], the following applies:

availableFlagVSP = 1 (I-188)

predFlagLXVSP = 1 (I-189)

refIdxLXVSP = i (I-190)

mvLXVSP = DispVec[xPb][yPb] (I-191)

When availableFlagVSP is equal to 1 the following applies:

– The derivation process for a depth or disparity sample array from a depth picture as specified in clause I.8.5.7 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, the disparity vector DispVec[xPb][yPb], the reference view order index RefViewIdx[xPb][yPb], and the variable partIdc equal to 2 as inputs, and the outputs are the array disparitySamples of size (nPbW)x(nPbH), and the flag horSplitFlag.

– For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = 0..nPbW − 1 and y = 0..nPbH − 1:

  – The variable SubPbArrayPartSizeVSP[xPb + x][yPb + y] is set equal to (horSplitFlag ? (8, 4) : (4, 8)).

  – For X in the range of 0 to 1, inclusive, the following applies:

    – The variables SubPbArrayPredFlagLXVSP, SubPbArrayMvLXVSP, and SubPbArrayRefIdxLXVSP are derived as follows:

SubPbArrayPredFlagLXVSP[xPb + x][yPb + y] = predFlagLXVSP (I-192)

SubPbArrayRefIdxLXVSP[xPb + x][yPb + y] = refIdxLXVSP (I-193)

SubPbArrayMvLXVSP[xPb + x][yPb + y] =
predFlagLXVSP ? (disparitySamples[x][y], 0) : (0, 0) (I-194)

    – When ChromaArrayType is not equal to 0, the derivation process for chroma motion vectors as specified in clause 8.5.3.2.9 is invoked with SubPbArrayMvLXVSP[xPb + x][yPb + y] as input, and the output is SubPbArrayMvCLXVSP[xPb + x][yPb + y].

I.8.5.3.3 Decoding process for inter prediction samples

I.8.5.3.3.1 General

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– the luma motion vectors mvL0 and mvL1,

– when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags, predFlagL0 and predFlagL1.

Outputs of this process are:

– an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ of luma prediction samples, where $nCbS_L$ is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

When DepthFlag is equal to 1, the following applies, for X in the range of 0 to 1, inclusive:

– The variable mvLX is set equal to (mvLX << 2).

– When ChromaArrayType is not equal to 0, the variable mvCLX is set equal to (mvCLX << 2).

The variable $nCbS_L$ is set equal to nCbS. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to nCbS / SubWidthC and the variable $nCbSh_C$ is set equal to nCbS / SubHeightC.

1. Let $predSamplesL0_L$ and $predSamplesL1_L$ be (nPbW)x(nPbH) arrays of predicted luma sample values and, when ChromaArrayType is not equal to 0, $predSamplesL0_{Cb}$, $predSamplesL1_{Cb}$, $predSamplesL0_{Cr}$, and $predSamplesL1_{Cr}$ be (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays of predicted chroma sample values.

2. For X in the range of 0 to 1, inclusive, when predFlagLX is equal to 1, the following applies:

   – When predFlagLX is equal to 1, the following applies:

     – If iv_res_pred_weight_idx[xCb][yCb] is not equal to 0, the bilinear sample interpolation and residual prediction process as specified in clause I.8.5.3.3.3 is invoked with the luma locations (xCb, yCb), (xBl, yBl), the size of the current luma coding block nCbS, the width and the height of the current luma prediction block nPbW and nPbH, the prediction list utilization flags predFlagL0 and predFlagL1, the reference indices refIdxL0 and refIdxL1, the motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the motion vectors mvCL0 and mvCL1, and the prediction list indication X as inputs, and the array $predSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$, as outputs.

     – Otherwise (iv_res_pred_weight_idx[xCb][yCb] is equal to 0), the following applies:

       – The reference picture consisting of an ordered two-dimensional array $refPicLX_L$ of luma samples and, when ChromaArrayType is not equal to 0, two ordered two-dimensional arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.3.3.2 with refIdxLX as input.

       – The array $predSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$ are derived by invoking the fractional sample interpolation process specified in clause 8.5.3.3.3 with the luma locations (xCb, yCb) and (xBl, yBl), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors mvLX and, when ChromaArrayType is not equal to 0, mvCLX, and the reference arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cr}$ as inputs.

3. Depending on the value of IlluCompFlag[xCb][yCb], the array $predSamples_L$ is derived as follows:

   – If IlluCompFlag[xCb][yCb] is equal to 0, the following applies:

     – The prediction samples inside the current luma prediction block, $predSamples_L[x_L + xBl][y_L + yBl]$ with $x_L = 0..nPbW - 1$ and $y_L = 0..nPbH - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width nPbW, the prediction block height nPbH, and the sample arrays $predSamplesL0_L$ and $predSamplesL1_L$, and the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1, and cIdx equal to 0 as inputs.

– Otherwise (IlluCompFlag[xCb][yCb] is equal to 1), the following applies:

  – The prediction samples inside the current luma prediction block, $predSamples_L$[x_L + xBl][y_L + yBl] with x_L = 0..nPbW − 1 and y_L = 0..nPbH − 1, are derived by invoking the illumination compensated sample prediction process specified in clause I.8.5.3.3.2, with the luma location (xCb, yCb), the size of the current luma coding block nCbS, the luma location (xBl, yBl), the width and the height of the current luma prediction block nPbW and nPbH, the sample arrays $predSamplesL0_L$ and $predSamplesL1_L$, the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1, mvL0, mvL1 and cIdx equal to 0 as inputs.

4. When ChromaArrayType is not equal to 0, depending on the values of IlluCompFlag[xCb][yCb] and nPbW, the arrays $predSamples_{Cb}$, and $predSamples_{Cr}$ are derived as follows:

   – If IlluCompFlag[xCb][yCb] is equal to 0 or nPbW is less than or equal to 8, the following applies:

     – The prediction samples inside the current chroma component Cb prediction block, $predSamples_{Cb}$[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC] with x_C = 0..nPbW / SubWidthC − 1 and y_C = 0..nPbH / SubHeightC − 1 , are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width nPbW set equal to nPbW / SubWidthC, the prediction block height nPbH set equal to nPbH / SubHeightC, the sample arrays $predSamplesL0_{Cb}$ and $predSamplesL1_{Cb}$, and the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1, and cIdx equal to 1 as inputs.

     – The prediction samples inside the current chroma component Cr prediction block, $predSamples_{Cr}$[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC] with x_C = 0..nPbW / SubWidthC − 1 and y_C = 0..nPbH / SubHeightC − 1 , are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width nPbW set equal to nPbW / SubWidthC, the prediction block height nPbH set equal to nPbH / SubHeightC, the sample arrays $predSamplesL0_{Cr}$ and $predSamplesL1_{Cr}$, and the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1, and cIdx equal to 2 as inputs.

   – Otherwise (IlluCompFlag[xCb][yCb] is equal to 1 and nPbW is greater than 8), the following applies:

     – The prediction samples inside the current chroma component Cb prediction block, $predSamples_{Cb}$[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC] with x_C = 0..nPbW / SubWidthC − 1 and y_C = 0..nPbH / SubHeightC − 1 , are derived by invoking the illumination compensated sample prediction process specified in clause I.8.5.3.3.2, with the luma location (xCb, yCb), the size of the current luma coding block nCbS, the chroma location (xBl / SubWidthC, yBl / SubHeightC), the width and the height of the current chroma prediction block (nPbW / SubWidthC) and (nPbH / SubHeightC), the sample arrays $predSamplesL0_{Cb}$ and $predSamplesL1_{Cb}$, the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1, mvCL0, mvCL1, and cIdx equal to 1 as inputs.

     – The prediction samples inside the current chroma component Cr prediction block, $predSamples_{Cr}$[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC] with x_C = 0..nPbW / SubWidthC − 1 and y_C = 0..nPbH / SubHeightC − 1 , are derived by invoking the illumination compensated sample prediction process specified in clause I.8.5.3.3.2, with the luma location (xCb, yCb), the size of the current luma coding block nCbS, the chroma location (xBl / SubWidthC, yBl / SubHeightC), the width and the height of the current chroma prediction block (nPbW / SubWidthC) and (nPbH / SubHeightC), the sample arrays $predSamplesL0_{Cr}$ and $predSamplesL1_{Cr}$, the variables predFlagL0, predFlagL1, refIdxL0, refIdxL1, mvCL0, mvCL1, and cIdx equal to 2 as inputs.

I.8.5.3.3.2 Illumination compensated sample prediction process

I.8.5.3.3.2.1 General

Inputs to this process are:

– a location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left sample of the current picture,

– the size of current luma coding block nCbS,

– a location (xBl, yBl) specifying the top-left sample of the current luma or chroma prediction block relative to the top-left sample of the current luma coding block,

– two variables nPbW and nPbH specifying the width and height of the current luma or chroma prediction block,

– two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1,

– two prediction list utilization flags predFlagL0 and predFlagL1,

– two reference indices refIdxL0 and refIdxL1,

– two motion vector mvL0 and mvL1,

– a colour component index cIdx.

Output of this process is an (nPbW)x(nPbH) array predSamples of prediction sample values.

The variables bitDepth, shift1, shift2, offset1, and offset2 are derived as follows:

$$\text{bitDepth} = (\text{ cIdx } == 0 \text{) ? BitDepth}_Y : \text{BitDepth}_C \quad \text{(I-195)}$$

$$\text{shift1} = \text{Max}(2, 14 - \text{bitDepth}) \quad \text{(I-196)}$$

$$\text{shift2} = \text{Max}(3, 15 - \text{bitDepth}) \quad \text{(I-197)}$$

$$\text{offset1} = 1 << (\text{shift1} - 1) \quad \text{(I-198)}$$

$$\text{offset2} = 1 << (\text{shift2} - 1) \quad \text{(I-199)}$$

The derivation process for illumination compensation mode availability and parameters as specified in clause I.8.5.3.3.2.2 is invoked with the luma location (xCb, yCb), the size of the current luma coding block nCbS, the prediction list utilization flags predFlagL0 and predFlagL1, the reference indices refIdxL0 and refIdxL1, the motion vectors mvL0 and mvL1, the variable bitDepth, and the variable cIdx as inputs, and the outputs are the flags puIcFlagL0 and puIcFlagL1, the variables icWeightL0 and icWeightL1, and the variables icOffsetL0 and icOffsetL1.

Depending on the value of predFlagL0 and predFlagL1, the prediction samples predSamples[x][y] with x = 0..(nPbW − 1) and y = 0..(nPbH − 1) are derived as follows:

– For X in the range of 0 to 1, inclusive, the following applies:

  – When predFlagLX is equal to 1, the following applies:

$$\text{clipPredVal} = \text{Clip3}(0, (1 << \text{bitDepth}) - 1, (\text{predSamplesLX}[x][y] + \text{offset1}) >> \text{shift1}) \quad \text{(I-200)}$$

$$\text{predValX} = \text{!puIcFlagLX ? clipPredVal :} \\ (\text{Clip3}(0, (1 << \text{bitDepth}) - 1, (\text{clipPredVal} * \text{icWeightLX}) >> 5) + \text{icOffsetLX}) \quad \text{(I-201)}$$

– If predFlagL0 and predFlagL1 are equal to 1, the following applies:

$$\text{predSamples}[x][y] = \text{Clip3}(0, (1 << \text{bitDepth}) - 1, (\text{predVal0} + \text{predVal1} + \text{offset2}) >> \text{shift2}) \quad \text{(I-202)}$$

– Otherwise (predFlagL0 is equal to 0 or predFlagL1 is equal to 0), the following applies:

$$\text{predSamples}[x][y] = \text{predFlagL0 ? predVal0 : predVal1} \quad \text{(I-203)}$$

I.8.5.3.3.2.2 Derivation process for illumination compensation mode availability and parameters

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– the size of the current luma coding block nCbS,

– two prediction list utilization flags predFlagL0 and predFlagL1,

– two reference indices refIdxL0 and refIdxL1,

– two motion vectors mvL0 and mvL1,

– a bit depth of samples bitDepth,

– a variable cIdx specifying colour component index.

Outputs of this process are:

– the flags puIcFlagL0 and puIcFlagL1 specifying whether illumination compensation is enabled,

– the variables icWeightL0 and icWeightL1 specifying weights for illumination compensation,

– the variables icOffsetL0 and icOffsetL1 specifying offsets for illumination compensation.

The variables puIcFlagL0 and puIcFlagL1 are set equal to 0, the variables icWeightL0 and icWeightL1 are set equal to 1, and the variables icOffsetL0 and icOffsetL1 are set equal to 0.

The variables subWidth, subHeight, and the location (xC, yC) specifying the top-left sample of the current luma or chroma

coding block are derived as follows:

$$\text{subWidth} = (\ \text{cIdx} \ == \ 0\ )\ ?\ 1 : \text{SubWidthC} \quad \text{(I-204)}$$

$$\text{subHeight} = (\ \text{cIdx} \ == \ 0\ )\ ?\ 1 : \text{SubHeightC} \quad \text{(I-205)}$$

$$(\ \text{xC}, \text{yC}\ ) = (\ \text{xCb} / \text{subWidth}, \text{yCb} / \text{subHeight}\ ) \quad \text{(I-206)}$$

The variable availFlagCurAboveRow specifying the availability of above neighbouring row samples is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring location (xN, yN) set equal to (xCb, yCb − 1) as inputs, and the output is assigned to availFlagCurAboveRow.

The variable availFlagCurLeftCol specifying the availability of left neighbouring column samples is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring location (xN, yN) set equal to (xCb − 1, yCb) as inputs, and the output is assigned to availFlagCurLeftCol.

When availFlagCurAboveRow is equal to 1 or availFlagCurLeftCol is equal to 1, the following applies:

1. Depending on the value of cIdx, the variable curRecSamples specifying the reconstructed picture samples before deblocking filter of the current picture is derived as follows:

$$\text{curRecSamples} = (\ \text{cIdx} \ == \ 0\ )\ ?\ S_L : (\ (\ \text{cIdx} \ == \ 1\ )\ ?\ S_{Cb} : S_{Cr}\ ) \quad \text{(I-207)}$$

2. For X in the range of 0 to 1, inclusive, when predFlagLX is equal to 1, the following applies:
   – Let refPicLX be the picture RefPicListX[refIdxLX].
   – When ViewIdx(refPicLX) is not equal to ViewIdx, the following applies:
     – The variable puIcFlagLX is set equal to 1.
     – The luma location (xRefBlkLX, yRefBlkLX) specifying the top-left sample of the reference block in the picture refPicLX is derived as follows:

$$\text{xRefBlkLX} = \text{xC} + (\ (\ \text{mvLX}[\ 0\ ] + (\ \text{cIdx}\ ?\ 4 : 2\ )\ ) \gg (\ 2 + (\ \text{cIdx}\ ?\ 1 : 0\ )\ )\ ) \quad \text{(I-208)}$$

$$\text{yRefBlkLX} = \text{yC} + (\ (\ \text{mvLX}[\ 1\ ] + (\ \text{cIdx}\ ?\ 4 : 2\ )\ ) \gg (\ 2 + (\ \text{cIdx}\ ?\ 1 : 0\ )\ )\ ) \quad \text{(I-209)}$$

     – Depending on the value of cIdx, the variable refRecSamplesLX specifying the reconstructed picture samples of the picture refPicLX is derived as follows:
       – If cIdx is equal to 0, refRecSamplesLX is set equal to reconstructed picture sample array S_L of picture refPicLX.
       – Otherwise, if cIdx is equal to 1, refRecSamplesLX is set equal to the reconstructed chroma sample array S_{Cb} of picture refPicLX.
       – Otherwise (cIdx is equal to 2), refRecSamplesLX is set equal to the reconstructed chroma sample array S_{Cr} of picture refPicLX.

3. The lists curSampleList, refSampleList0, and refSampleList1, specifying the neighbouring samples in pictures CurrPic, refPicL0, and refPicL1, respectively, are derived as follows:
   – The variable numSamples specifying the number of elements of curSampleList, refSampleList0, and refSampleList1 is set equal to 0.
   – For curNbColFlag in the range of 0 to 1, inclusive, the following applies:
     – When (curNbColFlag ? availFlagCurLeftCol : availFlagCurAboveRow) is equal to 1, the following applies, for i in the range of 0 to (nCbS / (curNbColFlag ? subHeight : subWidth)) − 1, inclusive:
       – The variables xOff and yOff are derived as follows:

$$\text{xOff} = \text{curNbColFlag}\ ?\ -1 : \text{i} \quad \text{(I-210)}$$

$$\text{yOff} = \text{curNbColFlag}\ ?\ \text{i} : -1 \quad \text{(I-211)}$$

       – For X in the range of 0 to 1, inclusive, when puIcFlagLX is equal to 1, the following applies:

$$\text{xP} = \text{Clip3}(\ 0, (\ \text{pic\_width\_in\_luma\_samples} / \text{subWidth}\ ) - 1, \text{xRefBlkLX} + \text{xOff}\ ) \quad \text{(I-212)}$$

$$\text{yP} = \text{Clip3}(\ 0, (\ \text{pic\_height\_in\_luma\_samples} / \text{subHeight}\ ) - 1, \text{yRefBlkLX} + \text{yOff}\ ) \quad \text{(I-213)}$$

$$\text{refSampleListLX[numSamples] = refRecSamplesLX[xP][yP]} \tag{I-214}$$

– The variables curSampleList and numSamples are modified as follows:

$$\text{curSampleList[numSamples++] = curRecSamples[xC + xOff][yC + yOff]} \tag{I-215}$$

4. For X in the range of 0 to 1, inclusive, when puIcFlagLX is equal to 1, icWeightLX and icOffsetLX are modified as follows:

   – The derivation process for illumination compensation parameters as specified in clause I.8.5.3.3.2.3 is invoked, with the list of neighbouring samples in the current picture curSampleList, the list of neighbouring samples in the reference picture refSampleListX, the number of neighbouring samples numSamples, the variable bitDepth, and the variable cIdx as inputs, and the variables icWeightLX and icOffsetLX as outputs.

I.8.5.3.3.2.3 Derivation process for illumination compensation parameters

Inputs to this process are:

– a list curSampleList specifying the current neighbouring samples,

– a list refSampleList specifying the reference neighbouring samples,

– a variable numSamples specifying the number of elements of curSampleList and refSampleList,

– a variable bitDepth specifying the bit depth of samples,

– a variable cIdx specifying colour component index.

Outputs of this process are:

– the variable icWeight specifying a weight for illumination compensation,

– the variable icOffset specifying an offset for illumination compensation.

The variable precShift is set equal to Max(0, bitDepth − 12).

The variables sumRef and sumCur are set equal to 0 and the following applies for i in the range of 0 to (numSamples / 2 − 1), inclusive:

$$\text{sumRef += refSampleList[2 * i]} \tag{I-216}$$

$$\text{sumCur += curSampleList[2 * i]} \tag{I-217}$$

The variables avgShift and avgOffset are derived as follows:

$$\text{avgShift = Log2(numSamples / 2)} \tag{I-218}$$

$$\text{avgOffset = 1 << (avgShift − 1)} \tag{I-219}$$

Depending on the value of cIdx, the variables icWeight and icOffset are derived as follows:

– If cIdx is equal to 0, the following applies:

  – The variables sumRefSquare and sumProdRefCur are set equal to 0, and the following applies for i in the range of 0 to (numSamples / 2 − 1), inclusive:

$$\text{sumRefSquare += (refSampleList[2 * i] * refSampleList[2 * i]) >> precShift} \tag{I-220}$$

$$\text{sumProdRefCur += (refSampleList[2 * i] * curSampleList[2 * i]) >> precShift} \tag{I-221}$$

  – The variables numerDiv and denomDiv are derived as follows:

$$\begin{aligned}\text{denomDiv} = &\text{((sumRefSquare + (sumRefSquare >> 7)) << avgShift)}\\ &\text{− ((sumRef * sumRef) >> precShift)}\end{aligned} \tag{I-222}$$

$$\begin{aligned}\text{numerDiv} = &\text{Clip3(0, 2 * denomDiv, ((sumProdRefCur + (sumRefSquare >> 7)) << avgShift)}\\ &\text{− ((sumRef * sumCur) >> precShift))}\end{aligned} \tag{I-223}$$

  – The variables shiftNumer and shiftDenom are derived as follows:

$$\text{shiftDenom = Max(0, Floor(Log2(Abs(denomDiv))) − 5)} \tag{I-224}$$

$$\text{shiftNumer = Max(0, shiftDenom − 12)} \tag{I-225}$$

  – The variables sNumerDiv and sDenomDiv are derived as follows:

$$\text{sDenomDiv = denomDiv >> shiftDenom} \tag{I-226}$$

sNumerDiv = numerDiv >> shiftNumer (I-227)

– The value of variable divCoeff is derived from Table I.3 depending on the value of sDenomDiv, and the variables icWeight and icOffset are derived as follows:

icWeight = (sNumerDiv * divCoeff) >> (shiftDenom − shiftNumer + 10) (I-228)

icOffset = (sumCur − ((icWeight * sumRef) >> 5) + avgOffset) >> avgShift (I-229)

– Otherwise (cIdx is not equal to 0), the following applies:

icWeight = 32 (I-230)

icOffset = (sumCur − sumRef + avgOffset) >> avgShift (I-231)

Table I.3 – Specification of divCoeff depending on sDenomDiv

sDenomDiv	0	1	2	3	4	5	6	7	8	9	10	11	12
divCoeff	0	32768	16384	10923	8192	6554	5461	4681	4096	3641	3277	2979	2731
sDenomDiv	13	14	15	16	17	18	19	20	21	22	23	24	25
divCoeff	2521	2341	2185	2048	1928	1820	1725	1638	1560	1489	1425	1365	1311
sDenomDiv	26	27	28	29	30	31	32	33	34	35	36	37	38
divCoeff	1260	1214	1170	1130	1092	1057	1024	993	964	936	910	886	862
sDenomDiv	39	40	41	42	43	44	45	46	47	48	49	50	51
divCoeff	840	819	799	780	762	745	728	712	697	683	669	655	643
sDenomDiv	52	53	54	55	56	57	58	59	60	61	62	63	
divCoeff	630	618	607	596	585	575	565	555	546	537	529	520	

I.8.5.3.3.3 Bilinear sample interpolation and residual prediction process

I.8.5.3.3.3.1 General

The process is only invoked when iv_res_pred_weight_idx[xCb][yCb] is in not equal to 0.

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the current luma prediction block,

– two prediction list utilization flags predFlagL0 and predFlagL1,

– two reference indices refIdxL0 and refIdxL1,

– two motion vectors mvL0 and mvL1,

– when ChromaArrayType is not equal to 0, two motion vectors mvCL0 and mvCL1,

– a prediction list indication X.

Outputs of this process are:

– the (nPbW)x(nPbH) array predSamplesLX_L,

– when ChromaArrayType is not equal to 0, the (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays $\text{predSamplesLX}_{Cb}$ and $\text{predSamplesLX}_{Cr}$.

The location (xPb, yPb) is derived as follows:

xPb = xCb + xBl (I-232)

yPb = yCb + yBl (I-233)

The prediction list indication variable Y is set equal to (1 − X) and the variable availFlag is set equal to 0.

The variable ivRefFlagLX is set equal to (DiffPicOrderCnt(CurrPic, RefPicListX[refIdxLX]) = = 0) and the variable ivRefFlagLY is set equal to predFlagLY ? (DiffPicOrderCnt(CurrPic, RefPicListY[refIdxLY]) = = 0) : 0.

Depending on the values of ivRefFlagLX, ivRefFlagLY, and RpRefIdxLX, the following applies:

– If ivRefFlagLX is equal to 0, the variable availFlag is set equal to 1, the variable refIdxLX is set equal to RpRefIdxLX, and the residual prediction motion vector scaling process as specified in clause I.8.5.3.3.3.4 is invoked with the prediction list indication variable equal to X, the motion vector mvLX, and the picture RefPicListX[refIdxLX] as inputs, and the modified motion vector mvLX as output.

– Otherwise (when ivRefFlagLX is equal to 1), the following applies:

  – If predFlagLY is equal to 1 and ivRefFlagLY is equal to 0, the following applies:

    – The variable availFlag is set equal to 1.

    – The residual prediction motion vector scaling process as specified in clause I.8.5.3.3.3.4 is invoked with the prediction list indication variable equal to Y, the motion vector mvLY, and the picture RefPicListY[refIdxLY] as inputs, and the modified motion vector mvLY as output.

    – The motion vector mvT is set equal to mvLY and the prediction list indication variable Z is set equal to Y.

  – Otherwise (predFlagLY is equal to 0 or ivRefFlagLY is equal to 1), the following applies:

    – The variable W is set equal to (predFlagLY && ivRefFlagLY) ? 0 : X.

    – The derivation process for a motion vector from a reference block for residual prediction as specified in clause I.8.5.3.3.3.5 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, the picture RefPicListW[refIdxLW], and the motion vector mvLW as inputs, and the flag availFlag, the motion vector mvT, and the prediction list utilization variable Z as outputs.

    – When availFlag is equal to 0 and RpRefPicAvailFlagLW is equal to 1, availFlag is set equal to 1, mvT is set equal to (0, 0), and Z is set equal to W.

When ChromaArrayType is not equal to 0, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with mvLX as input, and the output being mvCLX.

The array $predSamplesLX_L$, and, when ChromaArrayType is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$ are derived as follows:

– The reference picture consisting of an ordered two-dimensional array $refPicLX_L$ of luma samples and, when ChromaArrayType is not equal to 0, two ordered two-dimensional arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$ of chroma samples, are derived by invoking the reference picture selection process as specified in clause 8.5.3.3.2 with refIdxLX as input.

– The array $predSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$, are derived by invoking the bilinear sample interpolation process specified in clause I.8.5.3.3.3.2 with the luma locations (xCb, yCb) and (xBl, yBl), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vector mvLX, and, when ChromaArrayType is not equal to 0, the motion vector mvCLX, the reference array $refPicLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$ as inputs.

When one of the following conditions is true, availFlag is set equal to 0:

– ivRefFlagLX is equal to 0 and RefRpRefAvailFlagLX[RefViewIdx[xPb][yPb]] is equal to 0.

– ivRefFlagLX is equal to 1 and RefRpRefAvailFlagLZ[ViewIdx(RefPicListX[refIdxLX])] is equal to 0.

When availFlag is equal to 1, the following applies:

– Depending on the value of ivRefFlagLX, the variables rpPic, rpRefPic, and mvRp are derived as follows:

  – If ivRefFlagLX is equal to 0, the following applies:

    – Let rpPic be the picture with PicOrderCnt(rpPic) equal to PicOrderCntVal and nuh_layer_id equal to ViewCompLayerId[RefViewIdx[xPb][yPb]][DepthFlag].

    – Let rpRefPic be the picture with PicOrderCnt(rpRefPic) equal to PicOrderCnt(RefPicListX[RpRefIdxLX]) and nuh_layer_id equal to ViewCompLayerId[RefViewIdx[xPb][yPb]][DepthFlag].

    – The variable mvRp is set equal to DispVec[xPb][yPb].

- Otherwise (ivRefFlagLX is equal to 1), the following applies:
  - Let rpPic be the picture RefPicListZ[RpRefIdxLZ].
  - Let rpRefPic be the picture with PicOrderCnt(rpRefPic) equal to PicOrderCnt(rpPic) and nuh_layer_id equal to ViewCompLayerId[ViewIdx(RefPicListX[refIdxLX])][DepthFlag].
  - The variable mvRp is set equal to mvT.

- When ChromaArrayType is not equal to 0, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with mvRp as input, and the output being mvRpC.
- The array $rpSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $rpSamplesLX_{Cb}$ and $rpSamplesLX_{Cr}$ are derived as follows:
  - Let the reference sample array $rpPicLX_L$ correspond to the decoded sample array S_L derived in clause 8.7 for the previously decoded picture rpPic.
  - When ChromaArrayType is not equal to 0, let the reference sample arrays $rpPicLX_{Cb}$ and $rpPicLX_{Cr}$ correspond to the decoded sample arrays S_{Cb} and S_{Cr}, respectively, derived in clause 8.7 for the previously decoded picture rpPic.
  - The array $rpSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $rpSamplesLX_{Cb}$ and $rpSamplesLX_{Cr}$ are derived by invoking the bilinear sample interpolation process specified in clause I.8.5.3.3.3.2 with the luma locations (xCb, yCb) and (xBl, yBl), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors mvLX equal to mvRp, and, when ChromaArrayType is not equal to 0, the motion vector mvCLX equal to mvRpC, the reference array $rpPicLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $rpPicLX_{Cb}$ and $rpPicLX_{Cr}$ as inputs.
- The array $rpRefSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $rpRefSamplesLX_{Cb}$ and $rpRefSamplesLX_{Cr}$ are derived as follows:
  - Let the reference sample array $rpRefPicLX_L$ correspond to the decoded sample array S_L derived in clause 8.7 for the previously decoded picture rpRefPic.
  - When ChromaArrayType is not equal to 0, let the reference sample arrays $rpRefPicLX_{Cb}$ and $rpRefPicLX_{Cr}$ correspond to the decoded sample arrays S_{Cb} and S_{Cr}, respectively, derived in clause 8.7 for the previously decoded picture rpRefPic.
  - The array $rpRefSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $rpRefSamplesLX_{Cb}$ and $rpRefSamplesLX_{Cr}$ are derived by invoking the bilinear sample interpolation process specified in clause I.8.5.3.3.3.2 with the luma locations (xCb, yCb) and (xBl, yBl), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vector mvLX equal to (mvLX + mvRp), and, when ChromaArrayType is not equal to 0, the motion vector mvCLX equal to (mvCLX + mvRpC), the reference arrays with $rpRefPicLX_L$, and, when ChromaArrayType is not equal to 0, the arrays $rpRefPicLX_{Cb}$ and $rpRefPicLX_{Cr}$ as inputs.
- The variable shiftVal is set equal to (iv_res_pred_weight_idx[xCb][yCb] − 1).
- The modified prediction samples $predSamplesLX_L$[x][y] with x = 0..(nPbW) − 1 and y = 0..(nPbH) − 1 are derived as follows:

$$predSamplesLX_L[\,x\,][\,y\,] = predSamplesLX_L[\,x\,][\,y\,] + ((rpSamplesLX_L[\,x\,][\,y\,] - rpRefSamplesLX_L[\,x\,][\,y\,]) >> shiftVal) \quad \text{(I-234)}$$

- When ChromaArrayType is not equal to 0 and nPbW is greater than 8, the following applies:
  - The modified prediction samples $predSamplesLX_{Cb}$[x][y] with x = 0..(nPbW / SubWidthC) − 1 and y = 0..(nPbH / SubHeightC) − 1 are derived as follows:

$$predSamplesLX_{Cb}[\,x\,][\,y\,] = predSamplesLX_{Cb}[\,x\,][\,y\,] + ((rpSamplesLX_{Cb}[\,x\,][\,y\,] - rpRefSamplesLX_{Cb}[\,x\,][\,y\,]) >> shiftVal) \quad \text{(I-235)}$$

  - The modified prediction samples $predSamplesLX_{Cr}$[x][y] with x = 0..(nPbW / SubWidthC) − 1 and y = 0..(nPbH / SubHeightC) − 1 are derived as follows:

$$predSamplesLX_{Cr}[\,x\,][\,y\,] = predSamplesLX_{Cr}[\,x\,][\,y\,] + ((rpSamplesLX_{Cr}[\,x\,][\,y\,] - rpRefamplesLX_{Cr}[\,x\,][\,y\,]) >> shiftVal) \quad \text{(I-236)}$$

I.8.5.3.3.3.2 Bilinear sample interpolation process

The specifications in clause 8.5.3.3.3.1 apply with the following modifications:

– All invocations of the process specified in clause 8.5.3.3.3.2 are replaced with invocations of the process specified in clause I.8.5.3.3.3.3 with chromaFlag equal to 0 as additional input.

– All invocations of the process specified in clause 8.5.3.3.3.3 are replaced with invocations of the process specified in clause I.8.5.3.3.3.3 with chromaFlag equal to 1 as additional input.

I.8.5.3.3.3.3 Bilinear luma and chroma sample interpolation process

Inputs to this process are:

– a location in full-sample units (xInt, yInt),

– a location offset in fractional-sample units (xFrac, yFrac),

– a sample reference sample array refPicLX,

– a flag chromaFlag.

Output of this process is a predicted sample value predSampleLX.

Depending on the value of chromaFlag, the following applies:

– If chromaFlag is equal 0, the following applies:

  – The variables xFrac and yFrac are set equal to (xFrac << 1) and (yFrac << 1), respectively.

  – The variables picWidth and picHeight are set equal to pic_width_in_luma_samples and pic_height_in_luma_samples, respectively.

– Otherwise (chromaFlag is equal to 1), the variables picWidth and picHeight are set equal to PicWidthInSamplesC and PicHeightInSamplesC, respectively.

In Figure 8-5, the positions labelled with upper-case letters $B_{i,j}$ within shaded blocks represent samples at full-sample locations inside the given two-dimensional array refPicLX of samples. These samples may be used for generating the predicted sample value predSampleLX. The locations ($xB_{i,j}$, $yB_{i,j}$) for each of the corresponding samples $B_{i,j}$ inside the given array refPicLX of samples are derived as follows:

$$xB_{i,j} = \text{Clip3}(\ 0, \text{picWidth} - 1, \text{xInt} + i\ ) \quad \text{(I-237)}$$

$$yB_{i,j} = \text{Clip3}(\ 0, \text{picHeight} - 1, \text{yInt} + j\ ) \quad \text{(I-238)}$$

The positions labelled with lower-case letters within un-shaded blocks represent samples at eighth-pel sample fractional locations. The location offset in fractional-sample units (xFrac, yFrac) specifies which of the generated samples at full-sample and fractional-sample locations is assigned to the predicted sample value predSampleLX. This assignment is as specified in Table 8-9, with xFracC, yFracC, and predSampeLX_C replaced by xFrac, yFrac, and predSampleLX, respectively. The output is the value of predSampleLX.

The variables shift1, shift2, and shift3 are derived as follows:

– The variable bitDepth is set equal to chromaFlag ? BitDepth_C : BitDepth_Y.

– The variable shift1 is set equal to Min(4, bitDepth − 8), the variable shift2 is set equal to 6, and the variable shift3 is set equal to Max(2, 14 − bitDepth).

Given the samples $B_{i,j}$ at full-sample locations ($xB_{i,j}$, $yB_{i,j}$), the samples $ab_{0,0}$ to $hh_{0,0}$ at fractional sample positions are derived as follows:

– The samples labelled $ab_{0,0}$, $ac_{0,0}$, $ad_{0,0}$, $ae_{0,0}$, $af_{0,0}$, $ag_{0,0}$, and $ah_{0,0}$ are derived by applying a 2-tap filter to the nearest integer position samples as follows:

$$ab_{0,0} = (\ 56 * B_{0,0} + 8 * B_{1,0}) \gg \text{shift1} \quad \text{(I-239)}$$

$$ac_{0,0} = (\ 48 * B_{0,0} + 16 * B_{1,0}) \gg \text{shift1} \quad \text{(I-240)}$$

$$ad_{0,0} = (\ 40 * B_{0,0} + 24 * B_{1,0}) \gg \text{shift1} \quad \text{(I-241)}$$

$$ae_{0,0} = (\ 32 * B_{0,0} + 32 * B_{1,0}\ ) \gg \text{shift1} \quad \text{(I-242)}$$

$$af_{0,0} = (\ 24 * B_{0,0} + 40 * B_{1,0}\ ) \gg \text{shift1} \quad \text{(I-243)}$$

$$ag_{0,0} = (\ 16 * B_{0,0} + 48 * B_{1,0}\ ) \gg \text{shift1} \quad \text{(I-244)}$$

$$ah_{0,0} = (\ 8 * B_{0,0} + 56 * B_{1,0}\ ) \gg \text{shift1} \quad \text{(I-245)}$$

– The samples labelled $ba_{0,0}$, $ca_{0,0}$, $da_{0,0}$, $ea_{0,0}$, $fa_{0,0}$, $ga_{0,0}$, and $ha_{0,0}$ are derived by applying a 2-tap filter to the nearest

integer position samples as follows:

$ba_{0,0} = (56 * B_{0,0} + 8 * B_{0,1}) >> shift1$ (I-246)

$ca_{0,0} = (48 * B_{0,0} + 16 * B_{0,1}) >> shift1$ (I-247)

$da_{0,0} = (40 * B_{0,0} + 24 * B_{0,1}) >> shift1$ (I-248)

$ea_{0,0} = (32 * B_{0,0} + 32 * B_{0,1}) >> shift1$ (I-249)

$fa_{0,0} = (24 * B_{0,0} + 40 * B_{0,1}) >> shift1$ (I-250)

$ga_{0,0} = (16 * B_{0,0} + 48 * B_{0,1}) >> shift1$ (I-251)

$ha_{0,0} = (8 * B_{0,0} + 56 * B_{0,1}) >> shift1$ (I-252)

– The samples labelled $bX_{0,0}$, $cX_{0,0}$, $dX_{0,0}$, $eX_{0,0}$, $fX_{0,0}$, $gX_{0,0}$, and $hX_{0,0}$ for X being replaced by b, c, d, e, f, g, and h, respectively, are derived by applying a 2-tap filter to the intermediate values $aX_{0,i}$ with i = 0..1 in the vertical direction as follows:

$bX_{0,0} = (56 * aX_{0,0} + 8 * aX_{0,1}) >> shift2$ (I-253)

$cX_{0,0} = (48 * aX_{0,0} + 16 * aX_{0,1}) >> shift2$ (I-254)

$dX_{0,0} = (40 * aX_{0,0} + 24 * aX_{0,1}) >> shift2$ (I-255)

$eX_{0,0} = (32 * aX_{0,0} + 32 * aX_{0,1}) >> shift2$ (I-256)

$fX_{0,0} = (24 * aX_{0,0} + 40 * aX_{0,1}) >> shift2$ (I-257)

$gX_{0,0} = (16 * aX_{0,0} + 48 * aX_{0,1}) >> shift2$ (I-258)

$hX_{0,0} = (8 * aX_{0,0} + 56 * aX_{0,1}) >> shift2$ (I-259)

I.8.5.3.3.3.4 Residual prediction motion vector scaling process

Inputs to this process are:

– a prediction list indication variable X,

– a motion vector mvLX,

– a reference picture (associated with the motion vector mvLX) refPicLX.

Output of this process is a scaled motion vector mvLX.

The motion vector mvLX is scaled as follows:

tx = (16384 + (Abs(td) >> 1)) / td (I-260)

distScaleFactor = Clip3(−4096, 4095, (tb * tx + 32) >> 6) (I-261)

mv = Clip3(−32768, 32767, Sign(distScaleFactor * mvLX) *
((Abs(distScaleFactor * mvLX) + 127) >> 8)) (I-262)

where td and tb are derived as:

td = Clip3(−128, 127, DiffPicOrderCnt(CurrPic, refPicLX)) (I-263)

tb = Clip3(−128, 127, DiffPicOrderCnt(CurrPic, RefPicListX[RpRefIdxLX])) (I-264)

I.8.5.3.3.3.5 Derivation process for a motion vector from a reference block for residual prediction

Inputs to this process are:

– a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– two variables nPbW and nPbH specifying the width and the height of the current luma prediction block,

– a reference picture refPic,

– a disparity vector dispVec.

Outputs of this process are:

– a flag availFlag,

– a motion vector mvT,

– a prediction list indication variable Y.

The flag availFlag is set equal to 0, the motion vector mvT is set equal to (0, 0), and the prediction list indication variable Y is set equal to 0.

The reference luma location (xRef, yRef) in refPic is derived as follows:

$$\text{xRefFull} = \text{xPb} + (\text{nPbW} >> 1) + ((\text{dispVec}[0] + 2) >> 2) \quad \text{(I-265)}$$

$$\text{yRefFull} = \text{yPb} + (\text{nPbH} >> 1) + ((\text{dispVec}[1] + 2) >> 2) \quad \text{(I-266)}$$

$$\text{xRef} = \text{Clip3}(0, \text{pic\_width\_in\_luma\_samples} - 1, (\text{xRefFull} >> 3) << 3) \quad \text{(I-267)}$$

$$\text{yRef} = \text{Clip3}(0, \text{pic\_height\_in\_luma\_samples} - 1, (\text{yRefFull} >> 3) << 3) \quad \text{(I-268)}$$

Let refPb be the luma prediction block covering the luma position (xRef, yRef) in the picture refPic.

The variable cuPredModeRef[x][y] is set equal to CuPredMode[x][y] of the picture refPic.

When cuPredModeRef[xRef][yRef] is equal to MODE_SKIP or MODE_INTER, the following applies for X in the range of 0 to 1, inclusive:

– The variables predFlagRef[x][y], mvRef[x][y], and refIdxRef[x][y] are set equal to PredFlagLX[x][y], MvLX[x][y], and RefIdxLX[x][y], respectively, of picture refPic.

– The variable refPicListRef is set equal to RefPicListX of the slice containing refPb in the picture refPic.

– When availFlag is equal to 0, predFlagRef[xRef][yRef] is equal to 1, DiffPicOrderCnt(refPic, refPicListRef[refIdxRef[xRef][xRef]]) is not equal to 0, and RpRefPicAvailFlagLX is equal to 1, the following applies:

  – The variable availFlag is set equal to 1.

  – The variable Y is set equal to X.

  – The residual prediction motion vector scaling process as specified in clause I.8.5.3.3.3.4 is invoked with the prediction list indication variable equal to X, the motion vector mvRef[xRef][yRef], and the reference picture refPicListRef[refIdxRef[xRef][xRef]] as inputs, and the output being the motion vector mvT.

I.8.5.3.4 Decoding process for inter sample prediction for rectangular sub-block partitions

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block.

Outputs of this process are:

– an $(\text{nCbS}_L)\text{x}(\text{nCbS}_L)$ array predSamples_L of luma prediction samples, where nCbS_L is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(\text{nCbSw}_C)\text{x}(\text{nCbSh}_C)$ array predSamples_{Cb} of chroma prediction samples for the component Cb, where nCbSw_C and nCbSh_C are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(\text{nCbSw}_C)\text{x}(\text{nCbSh}_C)$ array predSamples_{Cr} of chroma prediction samples for the component Cr, where nCbSw_C and nCbSh_C are derived as specified below.

When ChromaArrayType is not equal to 0, the variable nCbSw_C is set equal to (nCbS / SubWidthC) and the variable nCbSh_C is set equal to (nCbS / SubHeightC).

The luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current picture is set equal to (xCb + xBl, yCb + yBl).

The variables nSbW and nSbH specifying the width and height of the sub-block partitions are derived as follows:

$$(\text{nSbW}, \text{nSbH}) = \text{SubPbArrayPartSize}[\text{xPb}][\text{yPb}] \quad \text{(I-269)}$$

For x in the range of 0 to (nPbW / nSbW − 1), inclusive, the following applies:

– For y in the range of 0 to (nPbH / nSbH − 1), inclusive, the following applies:

  – The luma location (xSb, ySb) specifying the top-left sample of the current luma sub-block partition relative to the top-left sample of the current luma coding block is derived as follows:

$$xSb = xBl + x * nSbW \tag{I-270}$$

$$ySb = yBl + y * nSbH \tag{I-271}$$

  – For X in the range of 0 to 1, inclusive, the variables mvLX, mvCLX, refIdxLX, and predFlagLX are derived as follows:

$$mvLX = SubPbArrayMvLX[xCb + xSb][yCb + ySb] \tag{I-272}$$

$$mvCLX = (ChromaArrayType \ != \ 0) ? SubPbArrayMvCLX[xCb + xSb][yCb + ySb] : (0, 0) \tag{I-273}$$

$$refIdxLX = SubPbArrayRefIdxLX[xCb + xSb][yCb + ySb] \tag{I-274}$$

$$predFlagLX = SubPbArrayPredFlagLX[xCb + xSb][yCb + ySb] \tag{I-275}$$

  – The decoding process for inter sample prediction as specified in clause I.8.5.3.3.1 is invoked with the luma coding block location (xCb, yCb), the luma prediction block location (xBl, yBl) equal to (xSb, ySb), the luma coding block size block nCbS, the luma prediction block width nPbW equal to nSbW, the luma prediction block height nPbH equal to nSbH, the luma motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1 as inputs, and the inter prediction samples that are an (nCbSL)x(nCbSL) array $predSamples_L$ of prediction luma samples, and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr, as outputs.

I.8.5.3.5 Decoding process for inter sample prediction for depth predicted sub-block partitions

I.8.5.3.5.1.1 General

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– two luma motion vectors mvL0 and mvL1,

– when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1,

– two reference indices refIdxL0 and refIdxL1,

– two prediction list utilization flags predFlagL0, and predFlagL1,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ of luma prediction samples, where $nCbS_L$ is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)x(nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to nCbS. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to nCbS / SubWidthC and the variable $nCbSh_C$ is set equal to nCbS / SubHeightC.

The decoding process for inter sample prediction as specified in clause I.8.5.3.3.1 is invoked with the luma coding block location (xCb, yCb), the luma prediction block location (xBl, yBl) set to (0, 0), the luma coding block size nCbS, the luma prediction block width nPbW equal to nCbS, the luma prediction block height nPbH equal to nCbS, the luma motion vectors mvL0 and mvL1, when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1, the reference indices refIdxL0 and refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1 as inputs, and the inter prediction samples (predSamples) that are an $(nCbS_L)x(nCbS_L)$ array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two $(nCbSw_C)x(nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr, as outputs.

The partition pattern contourPattern is derived as follows:

– The derivation process for a depth or disparity sample array from a depth picture as specified in clause I.8.5.7 is invoked with the luma location (xBlk, yBlk) equal to (xCb, yCb), the variable nBlkW equal to nCbS, the variable nBlkH equal to nCbS, the disparity vector dispVec equal to DispRefVec[xCb][yCb], the reference view order index refViewIdx equal to RefViewIdx[xCb][yCb], and the variable partIdc equal to 1 as inputs, and the output is the array refSamples of size (nCbS)x(nCbS).

– The variable threshVal is derived as follows:

$$\begin{aligned}\text{threshVal} = (\ &\text{refSamples}[\ 0\ ][\ 0\ ] + \text{refSamples}[\ 0\ ][\ \text{nCbS} - 1\ ] \\ &+ \text{refSamples}[\ \text{nCbS} - 1\ ][\ 0\ ] + \text{refSamples}[\ \text{nCbS} - 1\ ][\ \text{nCbS} - 1\ ]\ ) >> 2\end{aligned} \quad \text{(I-276)}$$

– For x = 0..nCbS − 1 and y = 0..nCbS − 1, the following applies:

$$\text{contourPattern}[\ x\ ][\ y\ ] = (\ \text{refSamples}[\ x\ ][\ y\ ] > \text{threshVal}\ ) \quad \text{(I-277)}$$

The array TempSamplesDbbp_L and, when ChromaArrayType is not equal to 0, the arrays $\text{TempSamplesDbbp}_{Cb}$ and $\text{TempSamplesDbbp}_{Cr}$ are modified as follows:

```
for( y = 0; y < nCbSL; y++ )
    for( x = 0; x < nCbSL; x++ )
        if( contourPattern[ x ][ y ] = = ( partIdx != contourPattern[ 0 ][ 0 ] ) ) {
            TempSamplesDbbpL[ x ][ y ] = predSamplesL[ x ][ y ]
            if( ChromaArrayType != 0 && ( x % SubWidthC ) = = 0 && ( y % SubHeightC ) = = 0 ) {
                xC = x / SubWidthC                                                    (I-278)
                yC = y / SubHeightC
                TempSamplesDbbpCb[ xC ][ yC ] = predSamplesCb[ xC ][ yC ]
                TempSamplesDbbpCr[ xC ][ yC ] = predSamplesCr[ xC ][ yC ]
            }
        }
```

When partIdx is equal to 1, the array predSamples_L and, when ChromaArrayType is not equal to 0, the arrays predSamples_{Cb} and predSamples_{Cr} are modified as follows:

– The derivation process for contour boundary filtered samples as specified in clause I.8.5.3.5.1.2 is invoked with, the luma coding block size nCbS_L, the current coding block width nCbSw equal to nCbS_L, the current coding block height nCbSh equal to nCbS_L, the partition pattern contourPattern, and the array predSamples of prediction samples equal to TempSamplesDbbp_L as inputs, and the output is assigned to the array predSamples_L of luma prediction samples.

– When ChromaArrayType is not equal to 0, the derivation process for contour boundary filtered samples as specified in clause I.8.5.3.5.1.2 is invoked with, the luma coding block size nCbS_L, the current coding block width nCbSw equal to nCbSw_C, the current coding block height nCbSh equal to nCbSh_C, the partition pattern contourPattern, and the array predSamples of prediction samples equal to $\text{TempSamplesDbbp}_{Cb}$ as inputs, and the output is assigned to the array predSamples_{Cb} of chroma prediction samples.

– When ChromaArrayType is not equal to 0, the derivation process for contour boundary filtered samples as specified in clause I.8.5.3.5.1.2 is invoked with, the luma coding block size nCbS_L, the current coding block width nCbSw equal to nCbSw, the current coding block height nCbSh equal to nCbSh_C, the partition pattern contourPattern, and the array predSamples of prediction samples equal to $\text{TempSamplesDbbp}_{Cr}$ as inputs, and the output is assigned to the array predSamples_{Cr} of chroma prediction samples.

I.8.5.3.5.1.2 Derivation process for contour boundary filtered samples

Inputs to this process are:

– a variable nCbS_L specifying the size of the current luma coding block,

– a variable nCbSw specifying the width of the current luma or chroma coding block,

– a variable nCbSh specifying the height of the current luma or chroma coding block,

– an $(\text{nCbS}_L)\text{x}(\text{nCbS}_L)$ partition pattern contourPattern,

– an (nCbSw)x(nCbSh) array predSamples prediction samples.

Output of this process is a modified (nCbSw)x(nCbSh) array predSamples of prediction samples.

The (nCbSw)x(nCbSh) array p is set equal to predSamples.

The variable xOff, yOff, nCbS_N, and n are derived as follows:

– If PartMode is equal to PART_Nx2N, xOff is set equal to 1, yOff is set equal to 0, $nCbS_N$ is set equal to nCbSw, and n is set equal to ($nCbS_L$ / nCbSw).

– Otherwise (PartMode is not equal to PART_Nx2N), xOff is set equal to 0, yOff is set equal to 1, $nCbS_N$ is set equal to nCbSh, and n is set equal to ($nCbS_L$ / nCbSh).

The values of predSamples are derived as follows:

```
for( y = 0; y < nCbSh; y++ )
    for( x = 0; x < nCbSw; x++ ) {
        filt = p[ x ][ y ]
        prevFlag = contourPattern[ Max( 0, n * ( x − xOff ) ) ][ Max( 0, n * ( y − yOff ) ) ]          (I-279)
        nextFlag = contourPattern[ Min( n * ( x + xOff ), nCbSL − 1 ) ][ Min( n * ( y + yOff ), nCbSL − 1 ) ]
        if( prevFlag != nextFlag )
            filt = ( p[ Max( 0, x − xOff ) ][ Max( 0, y − yOff ) ] + ( filt << 1 ) +
                    p[ Min( x + xOff, nCbSN − 1 ) ][ Min( y + yOff, nCbSN − 1 ) ] ) >> 2
        predSamples[ x ][ y ] = filt
    }
```

I.8.5.4 Decoding process for the residual signal of coding units coded in inter prediction mode

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block.

Outputs of this process are:

– an $(nCbS_L)$x$(nCbS_L)$ array $resSamples_L$ of luma residual samples, where $nCbS_L$ is derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)$x$(nCbSh_C)$ array $resSamples_{Cb}$ of chroma residual samples for the component Cb, where nCbSwC and nCbShC are derived as specified below,

– when ChromaArrayType is not equal to 0, an $(nCbSw_C)$x$(nCbSh_C)$ array $resSamples_{Cr}$ of chroma residual samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to 1 << log2CbSize. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS_L$/ SubWidthC and the variable $nCbSh_C$ is set equal to $nCbS_L$ / SubHeightC.

Let $resSamples_L$ be an $(nCbS_L)$x$(nCbS_L)$ array of luma residual samples and, when ChromaArrayType is not equal to 0, let $resSamples_{Cb}$ and $resSamples_{Cr}$ be two $(nCbSw_C)$x$(nCbSh_C)$ arrays of chroma residual samples.

– If DcOnlyFlag[xCb][yCb] is equal to 0, the following applies, depending on the value of rqt_root_cbf:

  – If rqt_root_cbf is equal to 0 or cu_skip_flag[xCb][yCb] is equal to 1, all samples of the $(nCbS_L)$x$(nCbS_L)$ array $resSamples_L$ and, when ChromaArrayType is not equal to 0, all samples of the two $(nCbSw_C)$x$(nCbSh_C)$ arrays $resSamples_{Cb}$ and $resSamples_{Cr}$ are set equal to 0.

  – Otherwise (rqt_root_cbf is equal to 1), the following ordered steps apply:

    1. The decoding process for luma residual blocks as specified in clause 8.5.4.2 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable nCbS set equal to $nCbS_L$, and the $(nCbS_L)$x$(nCbS_L)$ array $resSamples_L$ as inputs, and a modified version of the $(nCbS_L)$x$(nCbS_L)$ array $resSamples_L$ as output.

    2. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 1, the variable nCbSw set equal to $nCbSw_C$, the variable nCbSh set equal to $nCbSh_C$, and the $(nCbSw_C)$x$(nCbSh_C)$ array $resSamples_{Cb}$ as inputs, and a modified version of the $(nCbSw_C)$x$(nCbSh_C)$ array $resSamples_{Cb}$ as output.

    3. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 2, the variable nCbSw set equal to $nCbSw_C$, the variable nCbSh set equal to $nCbSh_C$, and the $(nCbSw_C)$x$(nCbSh_C)$ array $resSamples_{Cr}$ as inputs, and a modified version of the $(nCbSw_C)$x$(nCbSh_C)$ array $resSamples_{Cr}$ as output.

– Otherwise (DcOnlyFlag[xCb][yCb] is equal to 1), for x in the range of 0 to $nCbS_L - 1$, inclusive, and y in the range of 0 to $nCbS_L - 1$, inclusive, $resSamples_L$[x][y] is set equal to DcOffset[xCb][yCb][0].

NOTE – When DcOnlyFlag[xCb][yCb] is equal to 1, ChromaArrayType is equal to 0 in this version of this Specification.

I.8.5.5 Derivation process for a disparity vector for texture layers

I.8.5.5.1 General

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block.

The flag dvAvailFlag is set equal to 0 and the disparity vector dispVec is set equal to (0, 0).

The variable checkParallelMergeFlag is derived as follows:

– If one or more of the following conditions are true, checkParallelMergeFlag is set equal to 1:

  – CuPredMode[xCb][yCb] is equal to MODE_SKIP.

  – CuPredMode[xCb][yCb] is equal to MODE_INTER and merge_flag[xCb][yCb] is equal to 1.

– Otherwise, checkParallelMergeFlag is set equal to 0.

When slice_temporal_mvp_enabled_flag is equal to 1, the derivation process for a disparity vector from temporal neighbouring blocks as specified in clause I.8.5.5.2 is invoked with the luma location (xCb, yCb), and the variable nCbS as inputs, and the outputs are the flag dvAvailFlag, the disparity vector dispVec, and the reference view order index refViewIdx.

When dvAvailFlag is equal to 0, the following applies for i in the range of 0 to 1, inclusive:

1. The variable N is set equal to (i = = 0) ? A_1 : B_1.

2. The variable (xN, yN) is set equal to (i = = 0) ? (xCb − 1, yCb + nCbS − 1) : (xCb + nCbS − 1, yCb − 1).

3. The derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to the (xCb, yCb) and the luma location (xN, yN) as inputs, and the output is assigned to nbAvailFlag.

4. When CuPredMode[xN][yN] is equal to MODE_INTRA, nbAvailFlag is set equal to 0.

5. When all of the following conditions are true, nbAvailFlag is set equal to 0.

   – checkParallelMergeFlag is equal to 1,

   – (xCb >> Log2ParMrgLevel) is equal to (xN >> Log2ParMrgLevel),

   – (yCb >> Log2ParMrgLevel) is equal to (yN >> Log2ParMrgLevel).

6. The flag tPredNbAvailFlag is set equal to nbAvailFlag.

7. When N is equal to B_1 and ((yN >> CtbLog2SizeY) << CtbLog2SizeY) is less than ((yCb >> CtbLog2SizeY) << CtbLog2SizeY), tPredNbAvailFlag is set equal to 0.

8. The flag tPredNbDvAvailFlagN is set equal to 0.

9. For X in the range of 0 to 1, inclusive, the following applies:

   – When dvAvailFlag is equal to 0, nbAvailFlag is equal to 1, and PredFlagLX[xN][yN] is equal to 1, the following applies:

     – If DiffPicOrderCnt(RefPicListX[RefIdxLX[xN][yN]], CurrPic) is equal to 0, the following applies:

       refViewIdx = ViewIdx(RefPicListX[RefIdxLX[xN][yN]]) (I-280)

       dispVec = MvLXN[xN][yN] (I-281)

       dvAvailFlag = 1 (I-282)

     – Otherwise (DiffPicOrderCnt(RefPicListX[RefIdxLX[xN][yN]] , CurrPic) is not equal to 0), when tPredNbAvailFlag is equal to 1, tPredNbDvAvailFlagN is equal to 0, CuPredMode[xN][yN] is equal to MODE_SKIP, and IvMcFlag[xN][yN] is equal to 1, the following applies:

tPredNbDispVecN = DispRefVec[xN][yN] (I-283)

tPredNbRefViewIdxN = RefViewIdx[xN][yN] (I-284)

tPredNbDvAvailFlagN = 1 (I-285)

For i in the range of 0 to 1, inclusive, the following applies:

– The variable N is set equal to (i == 0) ? A_1 : B_1.

– When dvAvailFlag is equal to 0 and tPredNbDvAvailFlagN is equal to 1, the following applies:

dispVec = tPredNbDispVecN (I-286)

refViewIdx = tPredNbRefViewIdxN (I-287)

dvAvailFlag = 1 (I-288)

When dvAvailFlag is equal to 0, refViewIdx is set equal to DefaultRefViewIdx and dispVec is set equal to (0, 0).

Depending on the value of DepthRefEnabledFlag, the following applies:

– If DepthRefEnabledFlag is equal to 1, the following applies:

  – The derivation process for a depth or disparity sample array from a depth picture as specified in clause I.8.5.7 is invoked with the luma location (xBlk, yBlk) equal to (xCb, yCb), the variable nBlkW equal to nCbS, the variable nBlkH equal to nCbS, the disparity vector dispVec, the reference view order index refViewIdx, and the variable partIdc equal to 0 as inputs, and the array disparitySamples of size (nCbS)x(nCbS) as output.

  – The disparity vector dispRefVec is set equal to (disparitySamples[0][0], 0).

– Otherwise (DepthRefEnabledFlag is equal to 0), the disparity vector dispRefVec is set equal to dispVec.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for x = xCb..(xCb + nCbS − 1), y = yCb..(yCb + nCbS − 1):

DispVec[x][y] = dispVec (I-289)

DispRefVec[x][y] = dispRefVec (I-290)

RefViewIdx[x][y] = refViewIdx (I-291)

I.8.5.5.2 Derivation process for a disparity vector from temporal neighbouring blocks

Inputs to this process are:

– a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block.

Outputs of this process are:

– the availability flag dvAvailFlag,

– the disparity vector dispVec,

– the reference view order index refViewIdx.

The luma location (xRef, yRef) is derived as follows:

xRef = ((xCb + nCbS / 2) >> 4) << 4 (I-292)

yRef = ((yCb + nCbS / 2) >> 4) << 4 (I-293)

The flag dvAvailFlag is set equal to 0 and dispVec is set equal to (0, 0).

For i from 0 to NumDdvCandPics − 1, inclusive, the following ordered steps apply:

1. Let colPb the luma prediction block covering the luma location (xRef, yRef) in picture DdvCandPicList[i].

2. For X in the range of 0 to 1, inclusive, the following applies:

   – The variables candPredFlag[x][y], candRefIdx[x][y], and candMV[x][y] are set equal to the variables PredFlagLX[x][y], RefIdxLX[x][y], and MvLX[x][y] of the picture DdvCandPicList[i], respectively.

– The variable candPicRefPicList is set equal to RefPicListX of the slice containing colPb in the picture DdvCandPicList[i].

– When colPb is not coded in an intra prediction mode and candPredFlag[xRef][yRef] is equal to 1, the following applies:

  – The variable candRefViewIdx is set equal to ViewIdx(candPicRefPicList[candRefIdx [xRef][yRef]]).

  – When dvAvailFlag is equal to 0, candRefViewIdx is not equal to the ViewIdx(DdvCandPicList[i]), and there is a reference picture with ViewIdx equal to candRefViewIdx in RefPicList0 or RefPicList1, the following applies:

$$\text{refViewIdx} = \text{candRefViewIdx} \quad \text{(I-294)}$$

$$\text{dispVec} = \text{candMV}[\ \text{xRef}\ ][\ \text{yRef}\ ] \quad \text{(I-295)}$$

$$\text{dvAvailFlag} = 1 \quad \text{(I-296)}$$

I.8.5.6 Derivation process for a disparity vector for depth layers

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block.

When DispAvailFlag is equal to 1, the following assignments are made for x = xCb..(xCb + nCbS − 1) and y = yCb..(yCb + nCbS − 1):

$$\text{DispVec}[\ x\ ][\ y\ ] = (\ \text{DepthToDisparityB}[\ \text{DefaultRefViewIdx}\ ][\ 1 \ll (\ \text{BitDepth}_Y - 1\ )\ ],\ 0\ ) \quad \text{(I-297)}$$

$$\text{DispRefVec}[\ x\ ][\ y\ ] = \text{DispVec}[\ x\ ][\ y\ ] \quad \text{(I-298)}$$

$$\text{RefViewIdx}[\ x\ ][\ y\ ] = \text{DefaultRefViewIdx} \quad \text{(I-299)}$$

I.8.5.7 Derivation process for a depth or disparity sample array from a depth picture

Inputs to this process are:

– a luma location (xBlk, yBlk) specifying the top-left sample of the current luma block relative to the top-left luma sample of the current picture,

– two variables nBlkW and nBlkH specifying the width and the height of the luma block, respectively,

– a disparity vector dispVec,

– a reference view order index refViewIdx,

– a variable partIdc specifying a sub-block partitioning.

Outputs of this process are:

– an (nBlkW)x(nBlkH) array dSamples of depth or disparity values (depending on partIdc),

– a flag horSplitFlag (when partIdc is equal to 2).

Let refDepPic the picture in the current access unit with nuh_layer_id equal to ViewCompLayerId[refViewIdx][1].

Let refDepPels be the array of reconstructed luma samples S_L of the picture refDepPic. The luma location (xRefBlk, yRefBlk) of top-left luma sample of a block in refDepPels is derived as follows:

$$\text{xRefBlk} = \text{xBlk} + (\ (\ \text{dispVec}[\ 0\ ] + 2\ ) \gg 2\ ) \quad \text{(I-300)}$$

$$\text{yRefBlk} = \text{yBlk} + (\ (\ \text{dispVec}[\ 1\ ] + 2\ ) \gg 2\ ) \quad \text{(I-301)}$$

Depending on the value of partIdc, the variables nSubBlkW, nSubBlkH, and horSplitFlag are derived as follows:

– If partIdc is equal to 0, the variables nSubBlkW, nSubBlkH, and horSplitFlag are set equal to nBlkW, nBlkH, and 0, respectively.

– Otherwise, if partIdc is equal to 1, the variables nSubBlkW, nSubBlkH, and horSplitFlag are set equal to 1, 1, and 0, respectively.

– Otherwise (partIdc is equal to 2), the following applies:

– The variable minSubBlkSizeFlag is derived as follows:

$$\text{minSubBlkSizeFlag} = (\,\text{nBlkW} \,\%\, 8 \;!=\; 0\,) \;||\; (\,\text{nBlkH} \,\%\, 8 \;!=\; 0\,) \quad \text{(I-302)}$$

– Depending on the value of minSubBlkSizeFlag, the following applies:

  – If minSubBlkSizeFlag is equal to 1, the following applies:

$$\text{horSplitFlag} = (\,\text{nBlkH} \,\%\, 8 \;!=\; 0\,) \quad \text{(I-303)}$$

  – Otherwise (minSubBlkSizeFlag is equal to 0), the following applies:

$$\text{xP0} = \text{Clip3}(\,0,\ \text{pic\_width\_in\_luma\_samples} - 1,\ \text{xRefBlk}\,) \quad \text{(I-304)}$$

$$\text{yP0} = \text{Clip3}(\,0,\ \text{pic\_height\_in\_luma\_samples} - 1,\ \text{yRefBlk}\,) \quad \text{(I-305)}$$

$$\text{xP1} = \text{Clip3}(\,0,\ \text{pic\_width\_in\_luma\_samples} - 1,\ \text{xRefBlk} + \text{nBlkW} - 1\,) \quad \text{(I-306)}$$

$$\text{yP1} = \text{Clip3}(\,0,\ \text{pic\_height\_in\_luma\_samples} - 1,\ \text{yRefBlk} + \text{nBlkH} - 1\,) \quad \text{(I-307)}$$

$$\begin{aligned}\text{horSplitFlag} = (\,&(\,\text{refDepPels}[\,\text{xP0}\,][\,\text{yP0}\,] < \text{refDepPels}[\,\text{xP1}\,][\,\text{yP1}\,]\,)\\ &==\ (\,\text{refDepPels}[\,\text{xP1}\,][\,\text{yP0}\,] < \text{refDepPels}[\,\text{xP0}\,][\,\text{yP1}\,]\,)\,)\end{aligned} \quad \text{(I-308)}$$

– The variables nSubBlkW and nSubBlkH are derived as follows:

$$\text{nSubBlkW} = \text{horSplitFlag}\ ?\ 8 : 4 \quad \text{(I-309)}$$

$$\text{nSubBlkH} = \text{horSplitFlag}\ ?\ 4 : 8 \quad \text{(I-310)}$$

The array dSamples is derived as follows:

– For j in the range of 0 to (nBlkH / nSubBlkH − 1), inclusive, the following applies:

  – For i in the range of 0 to (nBlkW / nSubBlkW − 1), inclusive, the following applies:

    – The variable maxDep is set equal to −1 and modified as follows:

```
xSubBlkOff = i * nSubBlkW
ySubBlkOff = j * nSubBlkH
xP0 = Clip3( 0, pic_width_in_luma_samples − 1, xRefBlk + xSubBlkOff )
yP0 = Clip3( 0, pic_height_in_luma_samples − 1, yRefBlk + ySubBlkOff )
xP1 = Clip3( 0, pic_width_in_luma_samples − 1, xRefBlk + xSubBlkOff + nSubBlkW − 1 )     (I-311)
yP1 = Clip3( 0, pic_height_in_luma_samples − 1, yRefBlk + ySubBlkOff + nSubBlkH − 1 )
maxDep = Max( maxDep, refDepPels[ xP0 ][ yP0 ] )
maxDep = Max( maxDep, refDepPels[ xP0 ][ yP1 ] )
maxDep = Max( maxDep, refDepPels[ xP1 ][ yP0 ] )
maxDep = Max( maxDep, refDepPels[ xP1 ][ yP1 ] )
```

    – The values of the array dSamples are derived as follows:

```
for( yOff = 0; yOff < nSubBlkH; yOff++ )
    for( xOff = 0; xOff < nSubBlkW; xOff++ ) {
        x = xSubBlkOff + xOff
        y = ySubBlkOff + yOff
        if( partIdc  = =  1 )                                                  (I-312)
            dSamples[ x ][ y ] = maxDep
        else
            dSamples[ x ][ y ] = DepthToDisparityB[ refViewIdx ][ maxDep ]
    }
```

I.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause 8.6 apply.

I.8.7 In-loop filter process

The specifications in clause 8.7 apply.

I.9 Parsing process

I.9.1 General

The specifications in clause 9.1 apply with the following modifications.

– All references to the process specified in clause 9.3 are replaced with references to the process specified in clause I.9.3.

I.9.2 Parsing process for 0-th order Exp-Golomb codes

The specifications in clause 9.2 and all its subclauses apply.

I.9.3 CABAC parsing process for slice segment data

I.9.3.1 General

The specifications in clause 9.3.1 apply with the following modifications.

– All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12, respectively.

– All invocations of the process specified in clause 9.3.2 to 9.3.4 are replaced with invocations of the process specified in clause I.9.3.2 to I.9.3.4.

– All invocations of the process specified in clause 9.3.2.4 are replaced with invocations of the process specified in clause I.9.3.2.3.

– All invocations of the process specified in clause 9.3.2.6 are replaced with invocations of the process specified in clause I.9.3.2.5.

I.9.3.2 Initialization process

I.9.3.2.1 General

The specifications in clause 9.3.2.1 apply with the following modifications.

– All invocations of the process specified in clause 9.3.2.2 are replaced with invocations of the process specified in clause I.9.3.2.2.

– All invocations of the process specified in clause 9.3.2.5 are replaced with invocations of the process specified in clause I.9.3.2.4.

– All invocations of the process specified in clause 9.3.2.6 are replaced with invocations of the process specified in clause I.9.3.2.5.

I.9.3.2.2 Initialization process for context variables

The specifications in clause 9.3.2.2 apply with the following modifications.

– All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12.

– Table I.4 is appended to the end of Table 9-4.

– Table I.5 to Table I.14 are appended to the end of the clause.

Table I.4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

Syntax structure	Syntax element	ctxTable	initType		
			0	**1**	**2**
coding_unit( )	skip_intra_flag	Table I.5	0	1	2
intra_mode_ext( )	no_dim_flag	Table I.6	0	1	2
	depth_intra_mode_idx_flag	Table I.7	0	1	2
cu_extension( )	skip_intra_mode_idx	Table I.8	0	1	2
	dbbp_flag	Table I.9	0	1	2
	dc_only_flag	Table I.10	0	1	2
	iv_res_pred_weight_idx	Table I.11		0..2	3..5
	illu_comp_flag	Table I.12		0	1
depth_dcs( )	depth_dc_present_flag	Table I.13	0	1	2
	depth_dc_abs	Table I.14	0	1	2

Table I.5 – Values of initValue for skip_intra_flag ctxIdx

Initialization variable	ctxIdx of skip_intra_flag ctxIdx		
	0	**1**	**2**
initValue	185	185	185

Table I.6 – Values of initValue for no_dim_flag ctxIdx

Initialization variable	ctxIdx of no_dim_flag ctxIdx		
	0	**1**	**2**
initValue	154	141	155

Table I.7 – Values of initValue for depth_intra_mode_idx_flag ctxIdx

Initialization variable	ctxIdx of depth_intra_mode_idx_flag		
	0	**1**	**2**
initValue	154	154	154

Table I.8 – Values of initValue for skip_intra_mode_idx ctxIdx

Initialization variable	ctxIdx of skip_intra_mode_idx ctxIdx		
	0	**1**	**2**
initValue	137	137	137

Table I.9 – Values of initValue for dbbp_flag ctxIdx

Initialization variable	ctxIdx of dbbp_flag		
	0	**1**	**2**
initValue	154	154	154

Table I.10 – Values of initValue for dc_only_flag ctxIdx

Initialization variable	ctxIdx of dc_only_flag ctxIdx		
	0	**1**	**2**
initValue	154	154	154

Table I.11 – Values of initValue for iv_res_pred_weight_idx ctxIdx

Initialization variable	ctxIdx of iv_res_pred_weight_idx					
	0	**1**	**2**	**3**	**4**	**5**
initValue	162	153	162	162	153	162

Table I.12 – Values of initValue for illu_comp_flag ctxIdx

Initialization variable	ctxIdx of illu_comp_flag	
	0	**1**
initValue	154	154

Table I.13 – Values of initValue for depth_dc_present_flag ctxIdx

Initialization variable	ctxIdx of depth_dc_present_flag		
	0	**1**	**2**
initValue	0	0	64

Table I.14 – Values of initValue for depth_dc_abs ctxIdx

Initialization variable	ctxIdx of depth_dc_abs		
	0	**1**	**2**
initValue	154	154	154

I.9.3.2.3 Storage process for context variables and Rice parameter initialization states

The specifications in clause 9.3.2.4 apply with the following modifications

– All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12.

I.9.3.2.4 Synchronization process for context variables and Rice parameter initialization states

The specifications in clause 9.3.2.5 apply with the following modifications

– All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12.

I.9.3.2.5 Initialization process for the arithmetic decoding engine

The specifications in clause 9.3.2.6 apply.

I.9.3.3 Binarization process

I.9.3.3.1 General

The specifications in clause 9.3.3.1 apply with the following modifications.

– All references to processes clauses 9.3.3.2 to 9.3.3.11 are replaced with references to the processes specified in clauses I.9.3.3.2 and I.9.3.3.10 respectively.

– Table I.15 is appended to the end of Table 9-43.

Table I.15 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
coding_unit( )	skip_intra_flag	FL	cMax = 1
intra_mode_ext( )	no_dim_flag	FL	cMax = 1
	depth_intra_mode_idx_flag	FL	cMax = 1
	wedge_full_tab_idx	FL	cMax = NumWedgePattern[log2PbSize] − 1
cu_extension( )	skip_intra_mode_idx	TR	cMax = 3, cRiceParam = 0
	dbbp_flag	FL	cMax = 1
	dc_only_flag	FL	cMax = 1
	iv_res_pred_weight_idx	TR	cMax = 2, cRiceParam = 0
	illu_comp_flag	FL	cMax = 1
depth_dcs( )	depth_dc_present_flag	FL	cMax = 1
	depth_dc_abs	I.9.3.3.11	-
	depth_dc_sign_flag	FL	cMax = 1

I.9.3.3.2 Truncated Rice (TR) binarization process

The specifications in clause 9.3.3.2 apply.

I.9.3.3.3 k-th order Exp-Golomb (EGk) binarization process

The specifications in clause 9.3.3.3 apply.

I.9.3.3.4 Limited k-th order Exp-Golomb (EGk) binarization process

The specifications in clause 9.3.3.4 apply.

I.9.3.3.5 Fixed-length (FL) binarization process

The specifications in clause 9.3.3.5 apply.

I.9.3.3.6 Binarization process for part_mode

Inputs to this process are a request for a binarization for the syntax element part_mode, a luma location (xCb, yCb), specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, a variable log2CbSize specifying the current luma coding block size, and the variable partPredIdc indicating a reference partition mode.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element part_mode is specified in Table I.16 depending on the values of CuPredMode[xCb][yCb], log2CbSize, and partPredIdc.

Table I.16 – Binarization for part_mode

	part_mode	partPredIdc	PartMode	Bin string

CuPredMode [xCb][yCb]				log2CbSize = = MinCbLog2SizeY			
				0		1	
				amp_enabled_flag		log2CbSize > 3	
				0	1	0	1
MODE_INTRA	0	0	PART_2Nx2N	-	-	1	1
	1	0	PART_NxN	-	-	0	0
MODE_INTER	0	0, 1, 2	PART_2Nx2N	1	1	1	1
	1	0	PART_2NxN	01	011	01	01
	2	0	PART_Nx2N	00	001	00	001
	3	0	PART_NxN	-	-	-	000
	4	0	PART_2NxnU	-	0100	-	-
	5	0	PART_2NxnD	-	0101	-	-
	6	0	PART_nLx2N	-	0000	-	-
	7	0	PART_nRx2N	-	0001	-	-
	1	1	PART_2NxN	0	01	0	0
	2	1	PART_2NxnU	-	000	-	-
	3	1	PART_2NxnD	-	001	-	-
	1	2	PART_Nx2N	0	01	0	0
	2	2	PART_nLx2N	-	000	-	-
	3	2	PART_nRx2N	-	001	-	-

I.9.3.3.7 Binarization process for intra_chroma_pred_mode

The specifications in clause 9.3.3.8 apply.

I.9.3.3.8 Binarization process for inter_pred_idc

The specifications in clause 9.3.3.9 apply.

I.9.3.3.9 Binarization process for cu_qp_delta_abs

The specifications in clause 9.3.3.10 apply.

I.9.3.3.10 Binarization process for coeff_abs_level_remaining[ ]

The specifications in clause 9.3.3.11 apply.

I.9.3.3.11 Binarization process for depth_dc_abs

Input to this process is a request for a binarization for the syntax element depth_dc_abs.

Output of this process is the binarization of the syntax element.

The binarization of the syntax element depth_dc_abs is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

– The prefix value of depth_dc_abs, prefixVal, is derived as follows:

$$\text{prefixVal} = \text{Min}(\text{ depth\_dc\_abs, 3 })\qquad\text{(I-313)}$$

– The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with cMax = 3 and cRiceParam = 0.

When prefixVal is greater than 2, the suffix bin string is present and it is derived as follows:

– The suffix value of depth_dc_abs, suffixVal, is derived as follows:

$$\text{suffixVal} = \text{depth\_dc\_abs} - 3\qquad\text{(I-314)}$$

– The suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for suffixVal with the Exp-Golomb order k set equal to 0.

I.9.3.4 Decoding process flow

I.9.3.4.1 General

The specifications in clause 9.3.4.1 apply with the following modifications.

– All references to the process specified in clause 9.3.3 are replaced with references to the process specified in clause I.9.3.3.

– All invocations of the process specified in clause 9.3.4.2 are replaced with invocations of the process specified in clause I.9.3.4.2.

– All invocations of the process specified in clause 9.3.4.3 are replaced with invocations of the process specified in clause I.9.3.4.3.

I.9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag

I.9.3.4.2.1 General

The specifications in clause 9.3.4.2.1 apply with the following modifications:

– All references to the process specified in clause 9.3.4.2.2 are replaced by references to the process specified in clause I.9.3.4.2.2.

– Table I.17 is appended to the end of Table 9-48.

Table I.17 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	**1**	**2**	**3**	**4**	**>= 5**
skip_intra_flag	0	na	na	na	na	na
no_dim_flag	0	na	na	na	na	na
depth_intra_mode_idx_flag	0	na	na	na	na	na
wedge_full_tab_idx	bypass	bypass	bypass	bypass	bypass	bypass
skip_intra_mode_idx	0	bypass	bypass	na	na	na
dbbp_flag	0	na	na	na	na	na
dc_only_flag	0	na	na	na	na	na
iv_res_pred_weight_idx	0, 1 (clause I.9.3.4.2.2)	2	na	na	na	na
illu_comp_flag	0	na	na	na	na	na
depth_dc_present_flag	0	na	na	na	na	na
depth_dc_abs	0	0	0	bypass	bypass	bypass
depth_dc_sign_flag	bypass	0	0	0	0	0

I.9.3.4.2.2 Derivation process of ctxInc using left and above syntax elements

The specifications in clause 9.3.4.2.2 apply with the following modifications and additions:

– Table I.18 is appended to the end of Table 9-49.

Table I.18 – Specification of ctxInc using left and above syntax elements

Syntax element	condL	condA	ctxInc
iv_res_pred_weight_idx[x0][x0]	iv_res_pred_weight_idx[xNbL][yNbL]		(condL && availableL)

– The assignment of ctxInc for the syntax elements iv_res_pred_weight_idx[x0][y0] is specified in Table 9-49.

I.9.3.4.2.3 Derivation process of ctxInc for the syntax elements last_sig_coeff_x_prefix and last_sig_coeff_y_prefix

The specifications in clause 9.3.4.2.3 apply.

I.9.3.4.2.4 Derivation process of ctxInc for the syntax element coded_sub_block_flag

The specifications in clause 9.3.4.2.4 apply.

I.9.3.4.2.5 Derivation process of ctxInc for the syntax element sig_coeff_flag

The specifications in clause 9.3.4.2.5 apply.

I.9.3.4.2.6 Derivation process of ctxInc for the syntax element coeff_abs_level_greater1_flag

The specifications in clause 9.3.4.2.6 apply.

I.9.3.4.2.7 Derivation process of ctxInc for the syntax element coeff_abs_level_greater2_flag

The specifications in clause 9.3.4.2.7 apply.

I.9.3.4.3 Arithmetic decoding process

The specifications in clause 9.3.4.3 and all its subclauses apply with the following modifications:

– All references to the process specified in clause 9.3.4.2 are replaced with references to the process specified in clause I.9.3.4.2.

I.9.3.5 Arithmetic encoding process (informative)

The specifications in clause 9.3.5 and all its subclauses apply with the following modifications:

– All references to the process specified in clause 9.3.4.3 are replaced with references to the process specified in clause I.9.3.4.3.

I.10 Specification of bitstream subsets

The specifications in clause G.10 apply.

I.11 Profiles, tiers, and levels

I.11.1 Profiles

I.11.1.1 3D Main profile

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the 3D Main profile, the following applies:

– Let olsIdx be the OLS index of the OLS, the sub-bitstream subBitstream and the base layer sub-bitstream baseBitstream are derived as specified in clause F.11.3.

When vps_base_layer_internal_flag is equal to 1, the base layer sub-bitstream baseBitstream shall obey the following constraints:

– The base layer sub-bitstream baseBitstream shall be indicated to conform to the Main profile.

The sub-bitstream subBitstream shall obey the following constraints:

– All active VPSs shall have vps_num_rep_formats_minus1 in the range of 0 to 15, inclusive.

– All active SPSs for a layer with nuh_layer_id equal to i and DepthLayerFlag[i] equal to 0 in subBitstream shall have chroma_format_idc equal to 1 only.

– All active SPSs for a layer with nuh_layer_id equal to i and DepthLayerFlag[i] equal to 1 in subBitstream shall have chroma_format_idc equal to 0 only.

– All active SPSs for layers in subBitstream shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpcm_enabled_flag, explicit_rdpcm_enabled_flag, extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag, and cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.

– CtbLog2SizeY derived from all active SPSs for layers in subBitstream shall be in the range of 4 to 6, inclusive.

– All active PPSs for layers in subBitstream shall have log2_max_transform_skip_block_size_minus2 and chroma_qp_offset_list_enabled_flag, when present, equal to 0 only.

– ScalabilityId[j][smIdx] derived according to any active VPS shall be equal to 0 for any smIdx value not equal to 0 or 1 and for any value of j such that layer_id_in_nuh[j] is among layerIdListTarget that was used to derive subBitstream.

– All active VPSs shall have alt_output_layer_flag[olsIdx] equal to 0 only.

– When ViewOrderIdx[i] or DepthLayerFlag[i] derived according to any active VPS is greater than to 0 for the layer with nuh_layer_id equal to i in subBitstream, num_ref_loc_offsets shall be equal to 0 in each active PPS for that layer.

– When ViewOrderIdx[i] or DepthLayerFlag[i] derived according to any active VPS is greater than 0 for the layer with nuh_layer_id equal to i in subBitstream, the values of pic_width_in_luma_samples and pic_height_in_luma_samples in each active SPS for that layer shall be equal to the values of pic_width_in_luma_samples and pic_height_in_luma_samples, respectively, in each active SPS for all reference layers of that layer.

– For a layer with nuh_layer_id iNuhLId equal to any value included in layerIdListTarget that was used to derive subBitstream, the value of NumRefLayers[iNuhLId], which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 9.

– All active SPSs for layers in subBitstream shall have sps_range_extension_flag and sps_scc_entension_flag equal to 0 only.

– All active PPSs for layers in subBitstream shall have pps_range_extension_flag and pps_scc_entension_flag equal to 0 only.

– All active SPSs for layers in subBitstream shall have bit_depth_luma_minus8 equal to 0 only.

– All active SPSs for layers in subBitstream shall have bit_depth_chroma_minus8 equal to 0 only.

– All active PPSs for layers in subBitstream shall have colour_mapping_enabled_flag equal to 0 only.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.

– When an active PPS for any layer in subBitstream has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.

– The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit( ) data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.

– general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for any layer in subBitstream shall not be equal to 255 (which indicates level 8.5).

– The tier and level constraints specified for the 3D Main profile in clause I.11.2 shall be fulfilled.

– For any active VPS, ViewOrderIdx[i] shall be greater than ViewOrderIdx[j] for any values of i and j among layerIdListTarget that was used to derive subBitstream such that DepthLayerFlag[i] is equal to DepthLayerFlag[j] and i is greater than j.

– For any active VPS, LayerIdxInVps[i] shall be equal to LayerIdxInVps[j] + 1 for any values of i and j among layerIdListTarget that was used to derive subBitstream such that ViewOrderIdx[i] is equal to ViewOrderIdx[j], DepthLayerFlag[i] is equal to 1, and DepthLayerFlag[j] is equal to 0.

In the remainder of this clause and clause I.11.2, all syntax elements in the profile_tier_level( ) syntax structure refer to those in the profile_tier_level( ) syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the 3D Main profile is indicated as follows:

– If OpTid of the output operation point is equal to vps_max_sub_layer_minus1, the conformance is indicated by general_profile_idc being equal to 8 or general_profile_compatibility_flag[8] being equal to 1, and general_max_12bit_constraint_flag being equal to 1, general_max_10bit_constraint_flag being equal to 1, general_max_8bit_constraint_flag being equal to 1, general_max_422chroma_constraint_flag being equal to 1, general_max_420chroma_constraint_flag being equal to 1, general_max_monochrome_constraint_flag being equal to 0, general_intra_constraint_flag being equal to 0, and general_one_picture_only_constraint_flag being equal to 0, and general_lower_bit_rate_constraint_flag being equal to 1.

– Otherwise (OpTid of the output operation point is less than vps_max_sub_layer_minus1), the conformance is indicated by sub_layer_profile_idc[OpTid] being equal to 8 or sub_layer_profile_compatibility_flag[OpTid][8] being equal to 1, and sub_layer_max_12bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_10bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_8bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_422chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_420chroma_constraint_flag[OpTid] being equal to 1,

sub_layer_max_monochrome_constraint_flag[OpTid] being equal to 0, sub_layer_intra_constraint_flag[OpTid] being equal to 0, and sub_layer_one_picture_only_constraint_flag[OpTid] being equal to 0, and sub_layer_lower_bit_rate_constraint_flag[OpTid] being equal to 1.

I.11.2 Tiers and levels

The specification in sub-clause G.11.2 and its sub-clauses apply with the following modifications:

– "Multiview Main profile" is replaced by "3D Main profile"

I.11.3 Decoder capabilities

When a decoder conforms to any profile specified in Annex I, it shall also have the INBLD capability specified in clause F.11.1.

Clause F.11.2 specifies requirements for a decoder conforming to any profile specified in Annex I.

I.12 Byte stream format

The specifications in clause G.12 apply.

I.13 Hypothetical reference decoder

The specifications in clause G.13 apply.

I.14 Supplemental enhancement information

I.14.1 General

The specifications in clause G.14.1 apply.

I.14.2 SEI payload syntax

I.14.2.1 General SEI payload syntax

The specifications in clause G.14.2.1 apply.

I.14.2.2 Annex D, Annex F, and Annex G SEI message syntax for 3D high efficiency video coding

The specifications in clauses G.14.2.2 through G.14.2.7 apply.

I.14.2.3 Alternative depth information SEI message syntax

alternative_depth_info(payloadSize) {	**Descriptor**
alternative_depth_info_cancel_flag	u(1)
if(alternative_depth_info_cancel_flag = = 0) {	
depth_type	u(2)
if(depth_type = = 0) {	
num_constituent_views_gvd_minus1	ue(v)
depth_present_gvd_flag	u(1)
z_gvd_flag	u(1)
intrinsic_param_gvd_flag	u(1)
rotation_gvd_flag	u(1)
translation_gvd_flag	u(1)
if(z_gvd_flag)	
for(i = 0; i <= num_constituent_views_gvd_minus1 + 1; i++) {	
sign_gvd_z_near_flag[i]	u(1)
exp_gvd_z_near[i]	u(7)
man_len_gvd_z_near_minus1[i]	u(5)
man_gvd_z_near[i]	u(v)
sign_gvd_z_far_flag[i]	u(1)
exp_gvd_z_far[i]	u(7)

man_len_gvd_z_far_minus1[i]	u(5)
man_gvd_z_far[i]	u(v)
}	
if(intrinsic_param_gvd_flag) {	
prec_gvd_focal_length	ue(v)
prec_gvd_principal_point	ue(v)
}	
if(rotation_gvd_flag)	
prec_gvd_rotation_param	ue(v)
if(translation_gvd_flag)	
prec_gvd_translation_param	ue(v)
for(i = 0; i <= num_constituent_views_gvd_minus1 + 1; i++) {	
if(intrinsic_param_gvd_flag) {	
sign_gvd_focal_length_x[i]	u(1)
exp_gvd_focal_length_x[i]	u(6)
man_gvd_focal_length_x[i]	u(v)
sign_gvd_focal_length_y[i]	u(1)
exp_gvd_focal_length_y[i]	u(6)
man_gvd_focal_length_y[i]	u(v)
sign_gvd_principal_point_x[i]	u(1)
exp_gvd_principal_point_x[i]	u(6)
man_gvd_principal_point_x[i]	u(v)
sign_gvd_principal_point_y[i]	u(1)
exp_gvd_principal_point_y[i]	u(6)
man_gvd_principal_point_y[i]	u(v)
}	
if(rotation_gvd_flag)	
for(j = 0; j < 3; j++) /* row */	
for(k = 0; k < 3; k++) { /* column */	
sign_gvd_r[i][j][k]	u(1)
exp_gvd_r[i][j][k]	u(6)
man_gvd_r[i][j][k]	u(v)
}	
if(translation_gvd_flag) {	
sign_gvd_t_x[i]	u(1)
exp_gvd_t_x[i]	u(6)
man_gvd_t_x[i]	u(v)
}	
}	
}	
if(depth_type = = 1) {	
min_offset_x_int	se(v)
min_offset_x_frac	u(8)
max_offset_x_int	se(v)
max_offset_x_frac	u(8)
offset_y_present_flag	u(1)
if(offset_y_present_flag){	
min_offset_y_int	se(v)

min_offset_y_frac	u(8)
max_offset_y_int	se(v)
max_offset_y_frac	u(8)
}	
warp_map_size_present_flag	u(1)
if(warp_map_size_present_flag) {	
warp_map_width_minus2	ue(v)
warp_map_height_minus2	ue(v)
}	
}	
}	
}	

I.14.3 SEI payload semantics

I.14.3.1 General SEI payload semantics

The specifications in clause G.14.3.1 apply with the following modifications and additions:

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 161, 165, 167, 168, 177, 178, 179, 200 to 202, inclusive, and 205.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 168, inclusive, 176 to 181, inclusive, 200 to 202, inclusive, and 205.

The semantics and persistence scope for each SEI message specified in this annex are specified in the semantics specification for each particular SEI message in the subclauses of this clause.

NOTE – Persistence information for SEI messages specified in this annex is informatively summarized in Table I.19.

Table I.19 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Alternative depth information	Specified by the semantics of the SEI message

I.14.3.2 Annex D, Annex F, and Annex G SEI message semantics for 3D high efficiency video coding

I.14.3.2.1 General

The specifications of clause G.14.3.2 and its subclauses apply with the modifications specified in clause I.14.3.2.2.

I.14.3.2.2 Scalable nesting SEI message semantics for 3D high efficiency video coding

The specifications of clause G.14.3.2.2 apply with the following additions:

An SEI message that has payloadType equal to 181 (Alternative depth information) shall not be directly contained in a scalable nesting SEI message.

I.14.3.3 Alternative depth information SEI message semantics

The alternative depth information SEI message indicates that the decoded depth samples are interpreted as being of an alternative depth format. To discriminate different alternative depth formats, the depth_type syntax element is used.

Let currPic be a picture in the current access unit containing the alpha channel information SEI message. The information of the alternative depth information SEI message persists in output order until any of the following are true:

– A new CVS begins.

– The bitstream ends.

– A picture picB in an access unit containing an alternative depth information SEI message is output having PicOrderCnt(picB) greater than PicOrderCnt(currPic), where PicOrderCnt(picB) and PicOrderCnt(currPic) are the PicOrderCntVal values of picB and currPic, respectively, immediately after the invocation of the decoding process

for picture order count for picB.

alternative_depth_info_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous alternative depth information SEI message in output order. alternative_depth_info_cancel_flag equal to 0 indicates that alternative depth information follows.

depth_type identifies an alternative depth type according to Table I.20. depth_type equal to 0 indicates that global view and depth (GVD) information is present in this SEI message. depth_type equal to 1 indicates that the decoded depth samples can be used to derive a warp map and view synthesis can be performed by image-domain warping. Values of depth_type that are not listed in Table I.20 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore alternative depth information SEI messages that contain reserved values of depth_type.

Table I.20 – Interpretation of depth_type

Value	Description
0	Global view and depth (GVD)
1	Warp map

When the SEI message signals GVD information, the pictures of a view with ViewIdx greater than 0 contain samples of multiple subsampled pictures (subsequently denoted as constituent pictures), which are located in a spatial packing arrangement. This GVD information can be used after the decoder output to appropriately rearrange the samples in order to produce additional views that are appropriate for display or other purposes (which are outside the scope of this Recommendation | International Standard).

Layers that are referred to by this GVD information shall conform to the Main Profile, the Multiview Main profile, or the 3D Main profile, as specified in Annexes A, G, and I, respectively. The depth representation type is defined in the depth representation information SEI message that shall be always present together with the GVD information.

When the CVS contains an alternative depth information SEI message with depth_type equal to 0, and the first access unit in the CVS is an IRAP access unit, there shall be an alternative depth information SEI message with depth_type equal to 0 in the first access unit of the CVS. When GVD information is present, the following applies:

– The CVS shall represent two views.

– A texture layer and a depth layer shall both be present for the view with ViewIdx equal to 0.

– A texture layer shall be present and a depth layer may be present for the view with ViewIdx greater than 0.

– The value of NumDirectRefLayers[LayerIdxInVps[depLayerId]] shall be equal to 0 for any value of depLayerId among nuh_layer_id values of layers of the view with ViewIdx equal to 0 or the view with ViewIdx greater than 0.

– The pictures of the view with ViewIdx equal to 0 shall represent full resolution pictures.

– Each picture of the view with ViewIdx greater than 0 shall contain a packing arrangement of 1 to 4 sub-sampled pictures of constituent views. Such sub-sampled pictures are denoted as constituent pictures.

– All constituent pictures shall have a width and a height equal to (pic_width_in_luma_samples / 2) and (pic_height_in_luma_samples /2) in luma samples, respectively.

The variable i (with a value in the range of 1 to num_constituent_views_minus1 + 1, inclusive) in the alternative_depth_info( ) syntax structure is an index that is associated with the location of the constituent picture in the picture of the view with ViewIdx greater than 0, as specified in Table I.21. With regard to the remaining syntax elements in the alternative_depth_info( ) syntax structure, i equal to 0 refers to the parameters of the view with ViewIdx equal to 0 and i greater than 0 refers to the parameter of one of the constituent views.

NOTE 1 – The ViewIdx of the view with ViewIdx greater than 0 is not used for identification of constituent views in the context of this SEI message, but the variable i.

Table I.21 – Locations of the top-left luma samples of constituent pictures packed in a picture with ViewIdx greater than 0 relative to the top-left luma sample of this picture

Constituent picture index i	Location of the top-left luma sample in a picture with ViewIdx greater than 0
1	(0, 0)
2	(0, pic_height_in_luma_samples / 2)

3	(pic_width_in_luma_samples / 2, 0)
4	(pic_width_in_luma_samples / 2, pic_height_in_luma_samples / 2)

num_constituent_views_gvd_minus1 plus 1 specifies the number of constituent pictures packed into each picture of the view with ViewIdx greater than 0. num_constituent_views_gvd_minus1 shall be in the range of 0 to 3, inclusive.

depth_present_gvd_flag equal to 1 specifies that the depth layer for the view with ViewIdx greater than 0 is present and contains constituent pictures with a packing arrangement as described above. depth_present_gvd_flag equal to 0 specifies that the depth layer for the view with ViewIdx greater than 0 is not present.

Each constituent picture of a depth layer for the view with ViewIdx greater than 0 is associated with a constituent picture of the texture layer for the view with ViewIdx greater than 0 in the same relative location.

NOTE 2 – The following SEI message parameters can be used along with the decoded pictures of the depth layers to project samples from the view with ViewIdx equal to 0 into the co-ordinates of constituent views such that the reconstructed views can be generated by combining the projected samples and the samples from the constituent views.

The function binToFp(s, e, n, v) is specified as follows:

$$\text{binToFp}(\ s, e, n, v\ ) = (-1)^{s} * (\ e\ ==\ 0\ ?\ (\ 2^{-(\,30\,+\,v\,)} * n\ ) : (\ 2^{e\,-\,31} * (\ 1 + n \div 2^{v}\ )\ )\ ) \qquad \text{(I-315)}$$

NOTE 3 – The above specification is similar to what is found in IEC 60559:1989.

z_gvd_flag equal to 1 indicates the presence of the syntax elements sign_gvd_z_near_flag[i], exp_gvd_z_near[i], man_len_gvd_z_near_minus1[i], man_gvd_z_near[i], sign_gvd_z_far_flag[i], exp_gvd_z_far[i], man_len_gvd_z_far_minus1[i], and man_gvd_z_far[i], for i in the range of 0 to num_constituent_views_minus1 + 1, inclusive. z_gvd_flag equal to 0 indicates that these syntax elements are not present.

intrinsic_param_gvd_flag equal to 1 indicates the presence of intrinsic camera parameter syntax elements. intrinsic_param_gvd_flag equal to 0 indicates that these syntax elements are not present.

rotation_gvd_flag equal to 1 indicates the presence of rotation camera parameter syntax elements. rotation_gvd_flag equal to 0 indicates that these syntax elements are not present. When rotation_gvd_flag is equal to 0, a default rotation camera parameter of a unit matrix value is inferred.

translation_gvd_flag equal to 1 indicates the presence of horizontal translation camera parameter syntax elements. translation_gvd_flag equal to 0 indicates that these syntax elements are not present.

sign_gvd_z_near_flag[i] equal to 0 indicates that the sign of the nearest depth value of the i-th camera is positive. sign_gvd_z_near_flag[i] equal to 1 indicates that the sign of the nearest depth value of the i-th camera is negative.

exp_gvd_z_near[i] specifies the exponent part of the nearest depth value of the i-th camera. The value of exp_gvd_z_near[i] shall be in the range of 0 to 126, inclusive. The value 127 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_z_near[i] is equal to 127, the value of zNear[i] is unspecified.

man_len_gvd_z_near_minus1[i] plus 1 specifies the length in bits of the mantissa of the nearest depth value of the i-th camera. The value of man_len_gvd_z_near_minus1[i] shall be in the range of 0 to 31, inclusive.

man_gvd_z_near[i] specifies the mantissa part of the nearest depth value of the i-th camera. The length of man_gvd_z_near[i] syntax elements is man_len_gvd_z_near_minus1[i] + 1 bits.

When exp_gvd_z_near[i] is not equal to 127, zNear[i] is set equal to binToFp(sign_gvd_z_near_flag[i], exp_gvd_z_near[i], man_gvd_z_near[i], man_len_gvd_z_near_minus1[i] + 1).

sign_gvd_z_far_flag[i] equal to 0 indicates that the sign of the farthest depth value of the i-th camera is positive. sign_gvd_z_far_flag[i] equal to 1 indicates that the sign of the farthest depth value of the i-th camera is negative.

exp_gvd_z_far[i] specifies the exponent part of the farthest depth value of the i-th camera. The value of exp_gvd_z_far[i] shall be in the range of 0 to 126, inclusive. The value 127 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_z_far[i] is equal to 127, the value of zFar[i] is unspecified.

man_len_gvd_z_far_minus1[i] plus 1 specifies the length in bits of the mantissa of the farthest depth value of the i-th camera. The value of man_len_gvd_z_far_minus1[i] shall be in the range of 0 to 31, inclusive.

man_gvd_z_far[i] specifies the mantissa part of the farthest depth value of the i-th camera. The length of man_gvd_z_far[i] syntax elements is man_len_gvd_z_far_minus1[i] + 1 bits.

When exp_gvd_z_far[i] is not equal to 127, zFar[i] is set equal to binToFp(sign_gvd_z_far_flag[i], exp_gvd_z_far[i], man_gvd_z_far[i], man_len_gvd_z_far_minus1[i] + 1).

prec_gvd_focal_length specifies the exponent of the maximum allowable truncation error for focalLengthX[i] and

focalLengthY[i] as given by $2^{-\text{prec\_gvd\_focal\_length}}$. The value of prec_gvd_focal_length shall be in the range of 0 to 31, inclusive.

prec_gvd_principal_point specifies the exponent of the maximum allowable truncation error for principalPointX[i] and principalPointY[i] as given by $2^{-\text{prec\_gvd\_principal\_point}}$. The value of prec_gvd_principal_point shall be in the range of 0 to 31, inclusive.

prec_gvd_rotation_param specifies the exponent of the maximum allowable truncation error for r[i][j][k] as given by $2^{-\text{prec\_gvd\_rotation\_param}}$. The value of prec_gvd_rotation_param shall be in the range of 0 to 31, inclusive.

prec_gvd_translation_param specifies the exponent of the maximum allowable truncation error for tX[i] as given by $2^{-\text{prec\_gvd\_translation\_param}}$. The value of prec_gvd_translation_param shall be in the range of 0 to 31, inclusive.

sign_gvd_focal_length_x[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the horizontal direction is positive. sign_gvd_focal_length_x[i] equal to 1 indicates that the sign of the focal length of the i-th camera in the horizontal direction is negative.

exp_gvd_focal_length_x[i] specifies the exponent part of the focal length of the i-th camera in the horizontal direction. The value of exp_gvd_focal_length_x[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_focal_length_x[i] is equal to 63, the value of focal length of the horizontal direction for the i-th camera is unspecified.

man_gvd_focal_length_x[i] specifies the mantissa part of the focal length of the i-th camera in the horizontal direction. The length v of the man_gvd_focal_length_x[i] syntax element is determined as follows:

– If exp_gvd_focal_length_x[i] is equal to 0, the length v is set equal to Max(0, prec_gvd_focal_length − 30).

– Otherwise (exp_gvd_focal_length_x[i] is in the range of 1 to 62, inclusive), the length v is Max(0, exp_gvd_focal_length_x[i] + prec_gvd_focal_length − 31).

When exp_gvd_focal_length_x[i] is not equal to 63, the variable focalLengthX[i] is set equal to binToFp(sign_gvd_focal_length_x[i], exp_gvd_focal_length_x[i], man_gvd_focal_length_x[i], v).

sign_gvd_focal_length_y[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the vertical direction is positive. sign_gvd_focal_length_y[i] equal to 1 indicates that the sign of the focal length of the i-th camera in the vertical direction is negative.

exp_gvd_focal_length_y[i] specifies the exponent part of the focal length of the i-th camera in the vertical direction. The value of exp_gvd_focal_length_y[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_focal_length_y[i] is equal to 63, the value of focal length of the vertical direction is unspecified.

man_gvd_focal_length_y[i] specifies the mantissa part of the focal length of the i-th camera in the vertical direction.

The length v of the man_gvd_focal_length_y[i] syntax element is determined as follows:

– If exp_gvd_focal_length_y[i] is equal to 0, the length v is set equal to Max(0, prec_gvd_focal_length − 30).

– Otherwise (exp_gvd_focal_length_y[i] is in the range of 1 to 62, inclusive), the length v is set equal to Max(0, exp_gvd_focal_length_y[i] + prec_gvd_focal_length − 31).

When exp_gvd_focal_length_y[i] is not equal to 63, the variable focalLengthY[i] is set equal to binToFp(sign_gvd_focal_length_y[i], exp_gvd_focal_length_y[i], man_gvd_focal_length_y[i], v).

sign_gvd_principal_point_x[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the horizontal direction is positive. sign_gvd_principal_point_x[i] equal to 1 indicates that the sign of the principal point of the i-th camera in the horizontal direction is negative.

exp_gvd_principal_point_x[i] specifies the exponent part of the principal point of the i-th camera in the horizontal direction. The value of exp_gvd_principal_point_x[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_principal_point_x[i] is equal to 63, the value of principal point in the horizontal direction for the i-th camera is unspecified.

man_gvd_principal_point_x[i] specifies the mantissa part of the principal point of the i-th camera in the horizontal direction The length v of the man_gvd_principal_point_x[i] syntax element in units of bits is determined as follows:

– If exp_gvd_principal_point_x[i] is equal to 0, the length v is set equal to Max(0, prec_gvd_principal_point − 30).

– Otherwise (exp_gvd_principal_point_x[i] is in the range of 1 to 62, inclusive), the length v is set equal to Max(0, exp_gvd_principal_point_x[i] + prec_gvd_principal_point − 31).

When exp_gvd_principal_point_x[i] is not equal to 63, the variable principalPointX[i] is set equal to

binToFp(sign_gvd_principal_point_x[i], exp_gvd_principal_point_x[i], man_gvd_principal_point_x[i], v).

sign_gvd_principal_point_y[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the vertical direction is positive. sign_gvd_principal_point_y[i] equal to 1 indicates that the sign of the principal point of the i-th camera in the vertical direction is negative.

exp_gvd_principal_point_y[i] specifies the exponent part of the principal point of the i-th camera in the vertical direction. The value of exp_gvd_principal_point_y[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_principal_point_y[i] is equal to 63, the value of principal point in the vertical direction for the i-th camera is unspecified.

man_gvd_principal_point_y[i] specifies the mantissa part of the principal point of the i-th camera in the vertical direction. The length v of the man_gvd_principal_point_y[i] syntax element in units of bits is determined as follows:

– If exp_gvd_principal_point_y[i] is equal to 0, the length v is set equal to Max(0, prec_gvd_principal_point − 30).

– Otherwise (exp_gvd_principal_point_y[i] is in the range of 1 to 62, inclusive), the length v is set equal to Max(0, exp_gvd_principal_point_y[i] + prec_gvd_principal_point − 31).

When exp_gvd_principal_point_y[i] is not equal to 63, the variable principalPointY[i] is set equal to binToFp(sign_gvd_principal_point_y[i], exp_gvd_principal_point_y[i], man_gvd_principal_point_y[i], v).

sign_gvd_r[i][j][k] equal to 0 indicates that the sign of (j, k) component of the rotation matrix for the i-th camera is positive. sign_gvd_r[i][j][k] equal to 1 indicates that the sign of (j, k) component of the rotation matrix for the i-th camera is negative.

exp_gvd_r[i][j][k] specifies the exponent part of (j, k) component of the rotation matrix for the i-th camera. The value of exp_gvd_r[i][j][k] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_r[i][j][k] is equal to 63, the value of rotation matrix is unspecified.

man_gvd_r[i][j][k] specifies the mantissa part of (j, k) component of the rotation matrix for the i-th camera.

The length v of the man_gvd_r[i][j][k] syntax element in units of bits is determined as follows:

– If exp_gvd_r[i] is equal to 0, the length v is set equal to Max(0, prec_gvd_rotation_param − 30).

– Otherwise (exp_gvd_r[i] is in the range of 1 to 62, inclusive), the length v is set equal to Max(0, exp_gvd_r[i] + prec_gvd_rotation_param − 31).

When exp_gvd_r[i][j][k] is not equal to 63, the variable r[i][j][k] is set equal to binToFp(sign_gvd_r[i][j][k], exp_gvd_r[i][j][k], man_gvd_r[i][j][k], v).

The rotation matrix R[i] for i-th camera is represented as follows:

$$\begin{bmatrix} r[i][0][0] & r[i][0][1] & r[i][0][2] \\ r[i][1][0] & r[i][1][1] & r[i][1][2] \\ r[i][2][0] & r[i][2][1] & r[i][2][2] \end{bmatrix} \tag{I-316}$$

sign_gvd_t_x[i] equal to 0 indicates that the sign of the horizontal component of the translation vector for the i-th camera is positive. sign_gvd_t_x[i] equal to 1 indicates that the sign of the horizontal component of the translation vector for the i-th camera is negative.

exp_gvd_t_x[i] specifies the exponent part of the horizontal component of the translation vector for the i-th camera. The value of exp_gvd_t_x[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_t_x[i] is equal to 63, the value of the translation vector is unspecified.

man_gvd_t_x[i] specifies the mantissa part of the horizontal component of the translation vector for the i-th camera.

The length v of the man_gvd_t_x[i] syntax element in units of bits is determined as follows:

– If exp_gvd_t_x[i] is equal to 0, the length v is set equal to Max(0, prec_gvd_translation_param − 30).

– Otherwise (exp_gvd_t_x[i] is in the range of 1 to 62, inclusive), the length v is set equal to Max(0, exp_gvd_t_x[i] + prec_gvd_translation_param − 31).

When exp_gvd_t_x[i] is not equal to 63, the variable tX[i] is set equal to binToFp(sign_gvd_t_x[i], exp_gvd_t_x[i], man_gvd_t_x[i], v).

When the SEI signals warp map information, syntax elements of this SEI can be used to derive a sparse set of positional correspondences between decoded pictures of different views from decoded pictures of depth layers.

min_offset_x_int, **min_offset_x_frac** specify the integer and the fractional part of the minimum offset for the horizontal direction of a warp map.

The variable minOffsetX is derived as follows:

$$\text{minOffsetX} = \text{min\_offset\_x\_int} + \text{min\_offset\_x\_frac} \div 256 \quad \text{(I-317)}$$

max_offset_x_int, **max_offset_x_frac** specify the integer and the fractional part of the maximum offset for the horizontal direction of a warp map.

The variable maxOffsetX value is derived as follows:

$$\text{maxOffsetX} = \text{max\_offset\_x\_int} + \text{max\_offset\_x\_frac} \div 256 \quad \text{(I-318)}$$

offset_y_present_flag equal to 1 specifies that min_offset_y_int, min_offset_y_frac, max_offset_y_int, and max_offset_y_frac are present. offset_y_present_flag equal to 0 specifies that min_offset_y_int, min_offset_y_frac, max_offset_y_int, and max_offset_y_frac are not present.

min_offset_y_int, **min_offset_y_frac** specify the integer and the fractional part of the minimum offset for the vertical direction of a warp map. When not present, the values of min_offset_y_int and min_offset_y_frac are inferred to be equal to 0.

The variable minOffsetY value is derived as follows:

$$\text{minOffsetY} = \text{min\_offset\_y\_int} + \text{min\_offset\_y\_frac} \div 256 \quad \text{(I-319)}$$

max_offset_y_int, **max_offset_y_frac** specify the integer and the fractional part of the maximum offset for the vertical direction of a warp map. When not present, the values of max_offset_y_int and max_offset_y_frac are inferred to be equal to 0.

The variable maxOffsetY value is derived as follows:

$$\text{maxOffsetY} = \text{max\_offset\_y\_int} + \text{max\_offset\_y\_frac} \div 256 \quad \text{(I-320)}$$

warp_map_size_present_flag equal to 1 specifies that a new warp map size is present, which is valid for the current and all following warp maps in output order until a new message with warp_map_size_present_flag equal to 1 is received or alternative_depth_info_cancel_flag is equal to 1. warp_map_size_present_flag equal to 0 specifies that the warp map size is not changed.

warp_map_width_minus2 plus 2 specifies the width of the warp map. The value of warp_map_width_minus2 shall be in the range of 0 to (pic_width_in_luma_samples − 2), inclusive. The variable warpMapWidth is set equal to (warp_map_width_minus2 + 2).

warp_map_height_minus2 plus 2 specifies the height of the warp map. The value of warp_map_height_minus2 shall be in the range of 0 to (pic_height_in_luma_samples >> offset_y_present_flag) − 2, inclusive. The variable warpMapHeight is set equal to (warp_map_height_minus2 + 2).

The variables deltaX, deltaY, scaleX, and scaleY are derived as follows:

$$\text{deltaX} = \text{pic\_width\_in\_luma\_samples} \div (\text{warpMapWidth} - 1) \quad \text{(I-321)}$$

$$\text{deltaY} = \text{pic\_height\_in\_luma\_samples} \div (\text{warpMapHeight} - 1) \quad \text{(I-322)}$$

$$\text{scaleX} = (\text{maxOffsetX} - \text{minOffsetX}) / ((1 \ll \text{BitDepth}_Y) - 1) \quad \text{(I-323)}$$

$$\text{scaleY} = (\text{maxOffsetY} - \text{minOffsetY}) / ((1 \ll \text{BitDepth}_Y) - 1) \quad \text{(I-324)}$$

Let recSamples[x][y] correspond to the reconstructed sample array S_L of a picture of a depth layer. The corresponding horizontal warp map component w[x][y][0] and the corresponding vertical warp map component w[x][y][1] for recSamples[x][y] are derived as follows:

```
for( x = 0; x < warpMapWidth; x++ )
    for( y = 0; y < warpMapHeight; y++ ) {
        w[ x ][ y ][ 0 ] = x * deltaX + minOffsetX + scaleX * recSamples[ x ][ y ]
        if( offset_y_present_flag )
            w[ x ][ y ][ 1 ] = y * deltaY + minOffsetY +
                        scaleY * recSamples[ x ][ y + pic_height_in_luma_samples / 2 ]        (I-325)
        else
            w[ x ][ y ][ 1 ] = y * deltaY
    }
```

A warp map w[x][y] derived using the reconstructed sample array S_L of a picture included in a particular access unit and in a depth layer of a particular view is associated with the pictures included in the particular view and in the particular

access unit. The warp map specifies a sparse set of positional correspondences. These correspondences identify semantically corresponding sample locations between pictures included in the particular view and the particular AU, and the pictures included in a neighbouring view and the particular AU as follows:

– If the warp map w[x][y] is associated with a picture of the leftmost view, the warp map specifies for each location (x * deltaX, y * deltaY) of this picture a corresponding location (2 * w[x][y][0], 2 * w[x][y][1]) in a picture of the closest neighbouring view on the right.

– Otherwise (the warp map w[x][y] is associated with a picture of a view different to the leftmost view), the warp map specifies for each location (x * deltaX, y * deltaY) in this picture a corresponding location (2 * w[x][y][0], 2 * w[x][y][1]) in a picture of the closest neighbouring view on the left.

I.15 Video usability information

The specifications in clause G.15 apply.

Bibliography

[1] Recommendation ITU-T H.222.0 (in force), *Information technology – Generic coding of moving pictures and associated audio information: Systems.*

ISO/IEC 13818-1(in force), *Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems.*

[2] Recommendation ITU-T H.264 (in force), Advanced video coding for generic audiovisual services.

ISO/IEC 14496-10: (in force), *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*.

[3] Recommendation ITU-T H.271 (in force), *Video back-channel messages for conveyance of status information and requests from a video receiver to a video sender*.

[4] Recommendation ITU-T H.320 (in force), Narrow-band visual telephone systems and terminal equipment.

[5] Recommendation ITU-T T.800 (in force), Information technology – JPEG 2000 image coding system: Core coding system.

ISO/IEC 15444-1 (in force), Information technology – JPEG 2000 image coding system: Core coding system.

[6] Recommendation ITU-R BT. 470-6 (1998), *Conventional television systems*.

[7] Recommendation ITU-R BT.601-7 (2011), *Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios*.

[8] Recommendation ITU-R BT.709-6 (2015), *Parameter values for the HDTV standards for production and international programme exchange*.

[9] Recommendation ITU-R BT.1358-0 (2007), *Studio parameters of 625 and 525 line progressive scan television systems*.

[10] Recommendation ITU-R BT.1358-1 (2007), *Studio parameters of 625 and 525 line progressive television systems*. (Historical.)

[11] Recommendation ITU-R BT.1361-0 (1998), *Worldwide unified colorimetry and related characteristics of future television and imaging systems*.

[12] Recommendation ITU-R BT.1700-0 (2005), *Characteristics of composite video signals for conventional analogue television systems*.

[13] Recommendation ITU-R BT.1886-0 (2011), *Reference electro-optical transfer function for flat panel displays used in HDTV studio production*.

[14] Recommendation ITU-R BT.2020-2 (2015), *Parameter values for ultra-high definition television systems for production and international programme exchange*.

[15] Recommendation ITU-R BT.2035 (2013), *A reference viewing environment for evaluation of HDTV program material or completed programmes*.

[16] Recommendation ITU-R BT.2100-2 (2018), *Image parameter values for high dynamic range television for use in production and international programme exchange.*

[17] ANSI/CTA 861-G (2016), *A DTV Profile for Uncompressed High Speed Digital Interfaces*.

[18] ARIB STD-B67 (2015), *Essential Parameter Values for the Extended Image Dynamic Range Television (EIDRTV) System for Programme Production*.

[19] ATSC A/341 (2019), *Video – HEVC.*

[20] CIE 15 (in force), *Colorimetry*.

[21] CEA 861.3 (2015), *HDR Static Metadata Extensions*.

[22] EBU Tech. 3213-E (1975), *EBU Standard for Chromaticity Tolerances for Studio Monitors*.

[23] IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems*.

[24] IEC 61966-2-1 (in force), *Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB.*

[25] IEC 61966-2-4 (in force), *Multimedia systems and equipment – Colour measurement and management – Part 2-4: Colour management – Extended-gamut YCC colour space for video applications – xvYCC.*

[26] Internet Engineering Task Force, RFC 3550, Standard 64 (in force), *RTP: A Transport Protocol for Real-Time Applications*.

[27] ISO 11664-3 (in force), *Colorimetry – Part 3: CIE tristimulus values*.

[28] ISO/IEC 14496-12: *Information technology – Coding of audio-visual objects – Part 12: ISO base media file format*.

[29] SMPTE EG 432-1 (2010), *Digital Source Processing – Color Processing for D-Cinema.*

[30] SMPTE RDD 5 (2006), *Film Grain Technology – Specifications for H.264/MPEG-4 AVC Bitstreams*.

[31] SMPTE RP 177 (1993), *Derivation of Basic Television Color Equations*.

[32] SMPTE RP 431-2 (2011), *D-Cinema Quality – Reference Projector and Environment, Annex C.*

[33] SMPTE RP 2050-1 (2012), *4:2:2 / 4:2:0 Format Conversion Minimizing Color Difference Signal Degradation in Concatenated Operations – Filtering*.

[34] SMPTE ST 12-1 (2014), *Time and Control Code*.

[35] SMPTE ST 170 (2004), *Television – Composite Analog Video Signal – NTSC for Studio Applications*.

[36] SMPTE ST 240 (1999), *Television – 1125-Line High-Definition Production Systems – Signal Parameters*.

[37] SMPTE ST 428-1 (2006), *D-Cinema Distribution Master (DCDM) – Image Characteristics*.

[38] SMPTE ST 2084 (2014), *High Dynamic Range Electro-Optical Transfer Function of Mastering Reference Displays*.

[39] SMPTE ST 2085 (2015), *$Y'D'_ZD'_X$ Color-Difference Computations for High Dynamic Range X'Y'Z' Signals*.

[40] SMPTE ST 2086 (2018), *Mastering Display Color Volume Metadata supporting High Luminance and Wide Color Gamut Images*.

[41] SMPTE ST 2113 (2019), *Colorimetry of P3 Color Spaces*.

[42] United States Federal Communications Commission (2003), *Title 47 Code of Federal Regulations 73.682 (a) (20)*.

[43] United States National Television System Committee (1953), *Recommendation for transmission standards for colour television*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2021